
ADAPTIVE DEEP DENSITY APPROXIMATION FOR FRACTIONAL FOKKER-PLANCK EQUATIONS

A PREPRINT

Li Zeng *

Xiaoliang Wan †

Tao Zhou ‡

October 27, 2022

ABSTRACT

In this work, we propose adaptive deep learning approaches based on normalizing flows for solving fractional Fokker-Planck equations (FPEs). The solution of a FPE is a probability density function (PDF). Traditional mesh-based methods are ineffective because of the unbounded computation domain, a large number of dimensions and the nonlocal fractional operator. To this end, we represent the solution with an explicit PDF model induced by a flow-based deep generative model, simplified KRnet, which constructs a transport map from a simple distribution to the target distribution. We consider two methods to approximate the fractional Laplacian. One method is the Monte Carlo approximation. The other method is to construct an auxiliary model with Gaussian radial basis functions (GRBFs) to approximate the solution such that we may take advantage of the fact that the fractional Laplacian of a Gaussian is known analytically. Based on these two different ways for the approximation of the fractional Laplacian, we propose two models, MCNF and GRBFNF, to approximate stationary FPEs and MCTNF to approximate time-dependent FPEs. To further improve the accuracy, we refine the training set and the approximate solution alternately. A variety of numerical examples is presented to demonstrate the effectiveness of our adaptive deep density approaches.

Keywords Fractional Fokker-Planck equation · Normalizing flow · Adaptive density approximation · Monte Carlo sampling · Gaussian radial basis functions

1 Introduction

The fractional Fokker-Planck equations (FPEs) describe the time evolution of the probability density function of particles driven by Levy noise as well as Gaussian noise. Compared to integer-order FPEs whose associated stochastic differential equations (SDEs) are only driven by Gaussian noise, the fractional FPEs have a much wider range of applications in physics, biology, and other fields [30, 8, 10] since more than one kind of noise are often needed to simulate complex systems in practice. However, it is very challenging to approximate the fractional FP equations due to the following four obstacles:

- (i) The solution is a probability density function requiring vanishing boundary, normality and non-negative conditions.
- (ii) The computational domain may be unbounded.
- (iii) The fractional Laplacian operator is nonlocal.

*LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing, China. Email: zengli@lsec.cc.ac.cn.

†Department of Mathematics and Center for Computation and Technology, Louisiana State University, Baton Rouge 70803, USA. Email: xlwan@lsu.edu.

‡LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing, China. Email: tzhou@lsec.cc.ac.cn.

- (iv) The problem may have a large number of dimensions.

Traditional methods such as finite difference method, finite element method, spectral method as well as path integral method [12, 1, 5, 42, 35] have been applied to approximate fractional FPEs. Most of these methods are limited to problems of dimension one or two because the mesh-based discretization of high-dimensional problems induces unaffordable computational cost. On the other hand, simulating the SDEs associated with the FPEs [39] needs a large number of sample paths. Thus more efficient methods are still needed to approximate the fractional FPEs.

Recently, deep learning techniques have shown strong vitality in solving PDEs, e.g. deep Galerkin method [31], deep Ritz method [9] and physics-informed neural networks (PINNs) [27]. These techniques have gained encouraging performance in many applications [28, 2, 17, 40, 37, 26, 43, 18, 23]. Meanwhile, many deep generative models such as generative adversarial networks (GANs) [14], variational autoencoder (VAE) [21] and normalizing flow (NF) [25, 29] have been successfully applied to learn forward and inverse SDEs [4, 44, 38, 22]. For instance, a physics-informed generative adversarial model was proposed in [36] to tackle high-dimensional SDEs. In [16], a normalizing field flow was developed to build surrogate models for uncertainty quantification problems. The key issue of these methods is to convert the PDE problem into an optimization problem constrained to physical laws where the loss function is discretized by random training points. The training points here refer to space-time collocation points where the equations are enforced through optimization. The choice of training points will significantly affect the final numerical accuracy especially for unbounded problems. An adaptive sampling procedure was proposed in [33, 11] to solve integer-order FPEs, where the training set is updated by the current approximate solution which will be subsequently improved by the new training set. We will employ a similar adaptive procedure to deal with fractional FPEs.

To alleviate the difficulties induced by the constraints of a probability density function (PDF) we consider an explicit PDF model given by the normalizing flow. A normalizing flow constructs an invertible mapping from a simple distribution to the target distribution and results in an explicit PDF through the change of variable. We represent the solution of the FPE via a normalizing flow. In particular, we employ KRnet [32], which has been successfully applied to estimate high-dimensional density function and to approximate integer-order FPEs [33, 11].

Since KRnet yields a PDF explicitly, the first difficulty is avoided naturally. What's more, as a generative model, KRnet can generate exact random samples efficiently, which resolves the second obstacle because the commonly used uniform samples cannot be applied to an unbounded domain and are not effective for a large truncated domain. Using KRnet, we may update the training points by new samples from the current KRnet which automatically generates more samples in the region of high density. It is well known that automatic differentiation brings great convenience to the approximation of PDEs. However, it only works for the computation of integer-order derivatives. An effective method is needed to tackle the fractional derivatives. Several approaches have been developed to discretize the fractional derivatives when the PDE solution is modeled by neural networks. For example, a finite difference method is applied in [24], and a directly Monte Carlo sampling approach was proposed in [15]. In [3], Gaussian radial basis functions (GRBFs) are used to represent the solution of fractional PDEs based on the fact that the fractional Laplacian of GRBFs can be derived analytically. In this work, we will employ the Monte Carlo sampling approach or auxiliary GRBFs to deal with the fractional Laplacian operator in the nonlocal FPEs.

Integrating the PDF model from KRnet, automatic differentiation for integer-order derivatives and Monte Carlo sampling/GRBFs approach for fractional Laplacian, we have developed two effective deep learning techniques to address the approximation of nonlocal FPEs without requiring any labeled data. Following are the main features of our approach:

- Our approach is based on the explicit PDF model given by KRnet, which satisfies naturally all the constraints of a PDF. This is different from work [41] which handles the constraints via adding penalty terms to the loss function.
- Our approach is an extension to the previous work [33, 11] where only FPEs with integer-order derivatives are investigated. We have paid particular attention to how to improve both the accuracy and efficiency when the fractional derivatives are involved.
- Being a machine learning scheme, the proposed approach is mesh-free and can be easily applied to high dimensional problems.

The remainder of this paper is structured as follows. In Section 2, we present a brief description of the fractional FPEs. Section 3 provides an adaptive density approximation scheme for stationary fractional FPEs. In Section 4, we generalize the approach to deal with time-dependent fractional FPEs. We demonstrate the effectiveness and efficiency of our adaptive sampling approach with several numerical experiments in Section 5 followed by some concluding remarks in Section 6.

2 Problem setup

The main aim of this work is to solve the fractional FPEs. We first give a brief introduction to the fractional FPEs.

2.1 Fractional Fokker-Planck equations

Consider the state variable \mathbf{X}_t modeled by the following stochastic differential equation

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t) dt + \boldsymbol{\sigma}(\mathbf{X}_t, t) d\mathbf{W}_t + d\mathbf{L}_t^\alpha, \quad (2.1)$$

where \mathbf{X}_t and $\boldsymbol{\mu}(\mathbf{X}_t, t)$ are d -dimensional random vectors, $\boldsymbol{\sigma}(\mathbf{X}_t, t)$ is a $d \times M$ matrix, \mathbf{W}_t is an M -dimensional standard Wiener process and \mathbf{L}_t^α is a α -stable Levy motion with $\alpha \in (0, 2)$. The probability density function (PDF) $p(\mathbf{x}, t)$ for \mathbf{X}_t satisfies the time-dependent FPE:

$$\frac{\partial p}{\partial t} = \mathcal{L}p - (-\Delta)^{\alpha/2}p, \quad (2.2)$$

where

$$\mathcal{L}p = -\nabla \cdot (p\boldsymbol{\mu}) + \frac{1}{2} \nabla \cdot \nabla \cdot (\boldsymbol{\sigma}\boldsymbol{\sigma}^T p), \quad (2.3)$$

is induced by the drift and the diffusion, and the following nonlocal Laplacian operator

$$(-\Delta)^{\alpha/2}p = C_{d,\alpha} \text{P.V.} \int_{\mathbb{R}^d \setminus \{0\}} \frac{p(\mathbf{x}) - p(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|_2^{d+\alpha}} d\mathbf{y}, \quad (2.4)$$

is induced by the Levy motion, where $|\cdot|_2$ indicates the ℓ_2 norm of a vector and P.V. denotes the principle value of the integral and $C_{d,\alpha}$ is a constant given by

$$C_{d,\alpha} = \frac{2^{\alpha-1} \alpha \Gamma(\frac{\alpha+d}{2})}{\pi^{d/2} \Gamma(1 - \alpha/2)}, \quad (2.5)$$

with $\Gamma(\cdot)$ being the gamma function.

In general, equation (2.2) is defined on \mathbb{R}^d with the following boundary condition

$$p(\mathbf{x}) \rightarrow 0 \quad \text{as} \quad |\mathbf{x}|_2 \rightarrow \infty. \quad (2.6)$$

Furthermore, the solution as a probability density function should be conservative and non-negative, i.e.,

$$\int_{\mathbb{R}^d} p(\mathbf{x}, t) d\mathbf{x} \equiv 1, \quad \text{and} \quad p(\mathbf{x}, t) \geq 0. \quad (2.7)$$

In this work, we first address the numerical approximation of equation (2.2) when $\partial_t p = 0$, i.e.,

$$(\mathcal{L} - (-\Delta)^{\alpha/2})p = 0, \quad (2.8)$$

and then consider the time-dependent FPE, i.e., $\partial_t p \neq 0$.

3 MCNF and GRBNF for stationary fractional FPE

3.1 A bird's-eye view of proposed approaches

As it is mentioned in Introduction, we resort to deep generative modeling to construct an explicit PDF model on \mathbb{R}^d to remove all the constraints of a PDF, which also alleviates the curse of dimensionality. Depending on how to approximate the fractional Laplacian operator, we will develop two approaches to solve the fractional FPE (see Table 1). In MCNF we approximate the fractional Laplacian by the Monte Carlo method while in GRBNF we introduce an auxiliary model to represent the approximate solution with Gaussian radial basis functions such that we may take advantage of the fact that the fractional Laplacian of a Gaussian is known explicitly. As for the time-dependent fractional FPEs, temporal KRnet is considered as in [11], see Section 4 for the definition of MCTNF.

Notations	methods
GRBFNF	Normalizing flow + Gaussian radial basis function
MCNF	Normalizing flow + Monte Carlo sampling
MCTNF	Temporal normalizing flow + Monte Carlo sampling

Table 1: NF indicates how to obtain a solution model. GRBF and MC indicate how to deal with the fractional Laplacian operator.

3.1.1 MCNF

Assume that the unknown PDF $p(\mathbf{x})$ is modeled by KRnet as $p_{\text{KRnet},\theta}$ which will be specified in Section 3.2. We adopt the idea of physics-informed neural network to deal with equation (2.8), where the overall residuals of equation (2.8) on some prescribed collocation points in the computation domain will be minimized. For the given training data $S = \{\mathbf{x}^i\}_{i=1}^{N_S}$, we define the following loss function,

$$L(p_{\text{KRnet},\theta}) := \frac{1}{N_S} \sum_{i=1}^{N_S} |R_{\theta}(\mathbf{x}^i)|^2, \quad (3.1)$$

where $R_{\theta}(\mathbf{x})$ is the residual

$$R_{\theta}(\mathbf{x}) := (\mathcal{L} - (-\Delta)^{\alpha/2})p_{\text{KRnet},\theta}(\mathbf{x}). \quad (3.2)$$

The optimal parameters θ^* is given by the following optimization problem

$$\theta^* = \arg \min_{\theta} L(p_{\text{KRnet},\theta}). \quad (3.3)$$

The stochastic approximation proposed in [15] is used to compute the fractional Laplacian of $p_{\text{KRnet},\theta}$, which will be specify in Section 3.3. Another key component of our approach is the adaptive improvement of $p_{\text{KRnet},\theta}$ (see Section 3.5), where the training set S is updated by samples from the current optimal model $p_{\text{KRnet},\theta^*}$ that will be subsequently improved by the new training set. When the convergence is reached, we expect that the samples in S are distributed in terms of the exact solution $p(\mathbf{x})$.

3.1.2 GRBFNF

We rewrite equation (2.8) as

$$\begin{cases} \mathcal{L}p_{\text{KRnet},\theta}(\mathbf{x}) = (-\Delta)^{\alpha/2}p_{\text{GRBF},\tilde{\theta}}(\mathbf{x}), \\ p_{\text{KRnet},\theta}(\mathbf{x}) = p_{\text{GRBF},\tilde{\theta}}(\mathbf{x}), \end{cases} \quad (3.4)$$

where $p_{\text{KRnet},\theta}(\mathbf{x})$ is the same as the model used for MCNF and $p_{\text{GRBF},\tilde{\theta}}(\mathbf{x})$ is an auxiliary model for $p(\mathbf{x})$ (see Section 3.4). In other words,

$$p(\mathbf{x}) \approx p_{\text{KRnet},\theta}(\mathbf{x}), \quad p(\mathbf{x}) \approx p_{\text{GRBF},\tilde{\theta}}(\mathbf{x}).$$

For a set $S = \{\mathbf{x}^i\}_{i=1}^{N_S}$ of collocations points on the computation domain, we consider the following optimization problem:

$$(\theta^*, \tilde{\theta}^*) = \arg \min_{\theta, \tilde{\theta}} \tilde{L}(p_{\text{KRnet},\theta}, p_{\text{GRBF},\tilde{\theta}}), \quad (3.5)$$

where the tuple $(\theta^*, \tilde{\theta}^*)$ is the minimizer of the loss function defined as

$$\begin{aligned} \tilde{L}(p_{\text{KRnet},\theta}, p_{\text{GRBF},\tilde{\theta}}) &= \frac{1}{N_S} \sum_{i=1}^{N_S} \left(\mathcal{L}p_{\text{KRnet},\theta}(\mathbf{x}^i) - (-\Delta)^{\alpha/2}p_{\text{GRBF},\tilde{\theta}}(\mathbf{x}^i) \right)^2 \\ &+ \frac{\beta_m}{N_S} \sum_{i=1}^{N_S} \left(p_{\text{KRnet},\theta}(\mathbf{x}^i) - p_{\text{GRBF},\tilde{\theta}}(\mathbf{x}^i) \right)^2, \end{aligned} \quad (3.6)$$

with $0 < \beta_m < \infty$ being a penalty parameter. The main difference of GRBFNF from MCNF is the introduction of the auxiliary model $p_{\text{GRBF},\tilde{\theta}}(\mathbf{x})$, which will be mainly used to simplify the computation of the fractional Laplacian. More specifically, $p_{\text{GRBF},\tilde{\theta}}(\mathbf{x})$ is a linear combination of the Gaussian radial basis functions with centers $\tilde{\mathbf{x}}_i \in S_{\text{center}}$, which corresponds to a neural network with one hidden layer. The fractional Laplacian of $p_{\text{GRBF},\tilde{\theta}}(\mathbf{x})$ can be computed efficiently because the fractional Laplacian of a standard Gaussian is known analytically.

3.2 The density model $p_{\text{KRnet},\theta}$

The constraints specified in equations (2.6) and (2.7) on $p(\mathbf{x})$ bring essential difficulties to mesh-based numerical schemes for the approximation of the fractional FPEs. To this end, we employ KRnet, a certain type of normalizing flow, to build an effective approximator for FPEs [11, 33].

Normalizing flows seek an invertible mapping that corresponds to a transport map between a specified distribution and an arbitrary one. Let $\mathbf{Z} \in \mathbb{R}^d$ be a simple reference random variable with a known PDF $p_{\mathbf{Z}}$, e.g., Gaussian. Let $f : \mathbf{x} \rightarrow \mathbf{z}$ be an invertible mapping defined by a normalizing flow. Then the PDF of $\mathbf{X} = f^{-1}(\mathbf{Z})$ is given by the change of variables, i.e.,

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f(\mathbf{x})) \left| \det \nabla_{\mathbf{x}} f(\mathbf{x}) \right|, \quad (3.7)$$

where $\nabla_{\mathbf{x}} f(\mathbf{x})$ is the Jacobian matrix. Given observations of \mathbf{X} , the unknown invertible mapping can be learned via the maximum likelihood estimations.

To construct a complex bijection f , a general idea is to stack a sequence of simple bijections, each of which is a shallow neural network, in other words, the overall mapping is a deep neural network. Namely, the mapping $f(\cdot)$ can be written in a composite form:

$$\mathbf{z} = f(\mathbf{x}) = f_{[L]} \circ f_{[L-1]} \circ \cdots \circ f_{[1]}(\mathbf{x}). \quad (3.8)$$

Its inverse and Jacobian determinants are given as

$$\mathbf{x} = f^{-1}(\mathbf{z}) = f_{[1]}^{-1} \circ \cdots \circ f_{[L-1]}^{-1} \circ f_{[L]}^{-1}(\mathbf{z}), \quad (3.9)$$

$$\left| \det \nabla_{\mathbf{x}} f(\cdot) \right| = \prod_{i=1}^L \left| \det \nabla_{\mathbf{x}_{[i-1]}} f_{[i]}(\cdot) \right|, \quad (3.10)$$

where $\mathbf{x}_{[i-1]}$ indicates the immediate variables with $\mathbf{x}_{[0]} = \mathbf{x}$, $\mathbf{x}_{[L]} = \mathbf{z}$. Many variants of f have been proposed to enhance the expressive power and alleviate the computational cost of Jacobian determinants at the same time [20, 6, 7]. Among them, a successful example is KRnet [7]. We here employ a simplified KRnet, which includes affine coupling layers with an invertible block-triangle structure and actnorm layers.

3.2.1 Actnorm layer: scale and bias layer

We adopt the Actnorm layer $L_{\text{Actn},[i]}$ with data dependent initialization proposed by Kingma and Dhariwal [20]:

$$\mathbf{y}_{[i]} = \mathbf{a}_i \odot \mathbf{x}_{[i]} + \mathbf{b}_i, \quad (3.11)$$

where \mathbf{a}_i and \mathbf{b}_i are trainable parameters. When data are available, the parameters \mathbf{b}_i and \mathbf{a}_i can be initialized by the statistical mean and standard deviation respectively from data. Otherwise, we may simply initialize \mathbf{b}_i and \mathbf{a}_i as $\mathbf{b}_i = \mathbf{0}$ and $\mathbf{a}_i = \mathbf{1}_d$, where $\mathbf{1}_d$ denotes a d -dimensional vector whose components are all 1. After initialization, the scale and bias are treated as regular trainable parameters that are independent of the data. The inverse can be easily obtained via

$$\mathbf{x}_{[i]} = (\mathbf{y}_{[i]} - \mathbf{b}_i) / \mathbf{a}_i, \quad (3.12)$$

where the division here is operated on each corresponding component.

3.2.2 Affine coupling layer

Let $\mathbf{x}_{[i]} = (\mathbf{x}_{[i],1}, \mathbf{x}_{[i],2})$ be a partition with $\mathbf{x}_{[i],1} \in \mathbb{R}^m$ and $\mathbf{x}_{[i],2} \in \mathbb{R}^{d-m}$. An affine coupling layer $L_{\text{Aff},[i]}(\cdot)$ is defined as

$$\begin{aligned} \mathbf{x}_{[i],1} &= \mathbf{x}_{[i-1],1}, \\ \mathbf{x}_{[i],2} &= \mathbf{x}_{[i-1],2} \odot (\mathbf{1}_{d-m} + \beta \tanh(\mathbf{s}_i(\mathbf{x}_{[i-1],1}))) + e^{\zeta_i} \odot \tanh(\mathbf{q}_i(\mathbf{x}_{[i-1],1})), \end{aligned} \quad (3.13)$$

where $|\beta| < 1$ is a user-specified parameter (a commonly used choice is $\beta = 0.6$), $\mathbf{s}_i, \mathbf{q}_i : \mathbb{R}^m \rightarrow \mathbb{R}^{d-m}$ are scaling and translation depending only on $\mathbf{x}_{[i-1],1}$, and $\zeta_i \in \mathbb{R}^{d-m}$ is a trainable variable. Notice that the inverse can be easily computed via:

$$\begin{aligned} \mathbf{x}_{[i-1],1} &= \mathbf{x}_{[i],1}, \\ \mathbf{x}_{[i-1],2} &= (\mathbf{x}_{[i],2} - e^{\zeta_i} \odot \tanh(\mathbf{q}_i(\mathbf{x}_{[i],1}))) \odot (\mathbf{1}_{d-m} + \beta \tanh(\mathbf{s}_i(\mathbf{x}_{[i],1})))^{-1}. \end{aligned} \quad (3.14)$$

The Jacobian of $\mathbf{x}_{[i]}(\cdot)$ is given by

$$\nabla_{\mathbf{x}_{[i-1]}} \mathbf{x}_{[i]}(\cdot) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \nabla_{\mathbf{x}_{[i-1],1}} \mathbf{x}_{[i],2} & \text{diag}(\mathbf{1}_{d-m} + \alpha \tanh(\mathbf{s}_i(\mathbf{x}_{[i-1],1}))) \end{bmatrix}. \quad (3.15)$$

Furthermore, we can model $\mathbf{s}_i, \mathbf{b}_i$ via neural networks

$$(\mathbf{s}_i, \mathbf{q}_i) = \text{NN}_{[i]}(\mathbf{x}_{[i-1],1}). \quad (3.16)$$

Note that $L_{\text{Aff},[i]}(\cdot)$ only changes $\mathbf{x}_{[i-1],2}$, implying that in the next affine coupling layer we should exchange the positions of $\mathbf{x}_{[i],1}$ and $\mathbf{x}_{[i],2}$ to ensure that each component of $\mathbf{x}_{[i]}$ will be updated.

Based on the actnorm layer and affine coupling layer, our simplified KRnet can be represented by

$$\mathbf{z} = f_{\text{KRnet}}(\mathbf{x}) = f_{[L]} \circ f_{[L-1]} \circ \cdots \circ f_{[1]}(\mathbf{x}), \quad (3.17)$$

$$f_{[i]} = L_{\text{Aff},[i]} \circ L_{\text{Actn},[i]}, \quad i = 1, \dots, L, \quad (3.18)$$

where $L_{\text{Aff},[i]}$ is an affine coupling layer defined by (3.13) and $L_{\text{Actn},[i]}$ is an Actnorm layer defined by (3.11).

3.3 Stochastic approximation of the fractional operators

To compute the fractional Laplacian of the $p_{\text{KRnet},\theta}(\mathbf{x})$ with $\alpha \in (0, 2)$, we apply the stochastic approximation proposed in [15].

Lemma 3.1. [15] *Given a function u , its fractional Laplacian can be decomposed over a neighborhood $B_{r_0}(\mathbf{x}) = \{\mathbf{y} \mid \|\mathbf{y} - \mathbf{x}\|_2 \leq r_0\}$ around \mathbf{x} and its complement as*

$$(-\Delta)^{\alpha/2} u(\mathbf{x}) = C_{d,\alpha} \left(\int_{\mathbf{y} \in B_{r_0}(\mathbf{x})} \frac{u(\mathbf{x}) - u(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|_2^{d+\alpha}} d\mathbf{y} + \int_{\mathbf{y} \notin B_{r_0}(\mathbf{x})} \frac{u(\mathbf{x}) - u(\mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|_2^{d+\alpha}} d\mathbf{y} \right). \quad (3.19)$$

which, if exists, takes the form

$$\begin{aligned} (-\Delta)^{\alpha/2} u(\mathbf{x}) &= C_{d,\alpha} \frac{|S^{d-1}| r_0^{2-\alpha}}{2(2-\alpha)} \mathbb{E}_{\boldsymbol{\xi} \sim \text{U}(S^{d-1}), r_1 \sim f_{\text{I}}(r)} \left[\frac{2u(\mathbf{x}) - u(\mathbf{x} - r_1 \boldsymbol{\xi}) - u(\mathbf{x} + r_1 \boldsymbol{\xi})}{r_1^2} \right] \\ &+ C_{d,\alpha} \frac{|S^{d-1}| r_0^{-\alpha}}{2\alpha} \mathbb{E}_{\boldsymbol{\eta} \sim \text{U}(S^{d-1}), r_2 \sim f_{\text{O}}(r)} [2u(\mathbf{x}) - u(\mathbf{x} - r_2 \boldsymbol{\eta}) - u(\mathbf{x} + r_2 \boldsymbol{\eta})]. \end{aligned} \quad (3.20)$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are uniformly distributed on the the unit $(d-1)$ -sphere S^{d-1} , $|S^{d-1}|$ denotes the surface area of S^{d-1} ,

$$f_{\text{I}}(r) = \frac{2-\alpha}{r_0^{2-\alpha}} r^{1-\alpha} \cdot \mathbf{1}_{r \in [0, r_0]}, \quad f_{\text{O}}(r) = \alpha r_0^\alpha r^{-1-\alpha} \mathbf{1}_{r \in [r_0, \infty)},$$

$\mathbf{1}_\Omega$ is a characteristic function and r_1 and r_2 can be sampled as

$$r_1/r_0 \sim \text{Beta}(2-\alpha, 1), \quad r_0/r_2 \sim \text{Beta}(\alpha, 1). \quad (3.21)$$

Notice that the first expectation in equation (3.20) may suffer the round-off error and give rise to numerical instability for an extremely small r . Therefore, the following approximation is considered in practice

$$\mathbb{E}_{\boldsymbol{\xi} \sim \text{U}(S^{d-1}), r_1 \sim f_{\text{I}}(r)} \left[\frac{2u(\mathbf{x}) - u(\mathbf{x} - r_1 \boldsymbol{\xi}) - u(\mathbf{x} + r_1 \boldsymbol{\xi})}{r_1^2} \right] \approx \mathbb{E}_{\boldsymbol{\xi} \sim \text{U}(S^{d-1}), r_1 \sim f_{\text{I}}(r)} \left[\frac{2u(\mathbf{x}) - u(\mathbf{x} - r_\epsilon \boldsymbol{\xi}) - u(\mathbf{x} + r_\epsilon \boldsymbol{\xi})}{r_\epsilon^2} \right], \quad (3.22)$$

with $r_\epsilon = \max\{\epsilon, r_1\}$, where $\epsilon > 0$ is a small positive number.

Combining the stochastic approximation for the fractional Laplacian operator and the physics-informed neural network (3.1), along with the automatic differentiation for the integer-order derivative, we obtain the final approximation for $L(p_{\text{KRnet},\theta})$ as follows

$$\begin{aligned} L(p_{\text{KRnet},\theta}) &\approx \hat{L}(p_{\text{KRnet},\theta}; r_\epsilon, r_0) \\ &= \frac{1}{N_S} \sum_{i=1}^{N_S} \left| -\nabla \cdot (\boldsymbol{\mu} p_{\text{KRnet},\theta})(\mathbf{x}^i) + \frac{1}{2} \nabla \cdot \nabla \cdot (\boldsymbol{\sigma} \boldsymbol{\sigma}^T p_{\text{KRnet},\theta})(\mathbf{x}^i) \right. \\ &\quad - C_{d,\alpha} \frac{|S^{d-1}| r_0^{2-\alpha}}{2(2-\alpha)} \mathbb{E}_{\boldsymbol{\xi} \sim \text{U}(S^{d-1}), r_1 \sim f_{\text{I}}(r)} \left[\frac{2p_{\text{KRnet},\theta}(\mathbf{x}^i) - p_{\text{KRnet},\theta}(\mathbf{x}^i - r_\epsilon \boldsymbol{\xi}) - p_{\text{KRnet},\theta}(\mathbf{x}^i + r_\epsilon \boldsymbol{\xi})}{r_\epsilon^2} \right] \\ &\quad \left. - C_{d,\alpha} \frac{|S^{d-1}| r_0^{-\alpha}}{2\alpha} \mathbb{E}_{\boldsymbol{\eta} \sim \text{U}(S^{d-1}), r_2 \sim f_{\text{O}}(r)} [2p_{\text{KRnet},\theta}(\mathbf{x}^i) - p_{\text{KRnet},\theta}(\mathbf{x}^i - r_2 \boldsymbol{\eta}) - p_{\text{KRnet},\theta}(\mathbf{x}^i + r_2 \boldsymbol{\eta})] \right|^2. \end{aligned} \quad (3.23)$$

In Lemma 3.1 we need samples from Beta distributions $\text{Beta}(2 - \alpha, 1)$ and $\text{Beta}(\alpha, 1)$. It is well known that $\text{Beta}(a, 1)$ becomes concentrated on origin as a goes to zero, see Fig. 1. Thus when α increases, the samples of r_1 in equation (3.22) may concentrate on the area close to zero, which indicates a bigger r_ϵ is needed to guarantee numerical stability.

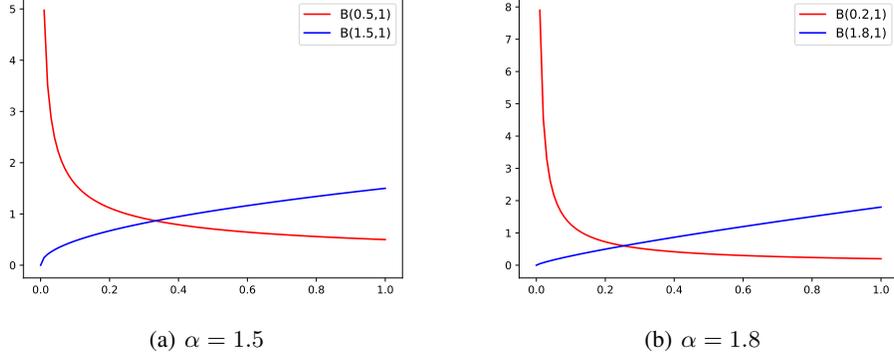


Figure 1: Beta distribution

3.4 The auxiliary density model $p_{\text{GRBF}, \tilde{\theta}}$

The definition of the auxiliary density model $p_{\text{GRBF}, \tilde{\theta}}$ is based on the following lemma [3]:

Lemma 3.2. *Let u be a Gaussian function of the form $u(\mathbf{x}) = \exp(-\sigma^{-2}|\mathbf{x} - \mathbf{x}_0|_2^2)$ for $\mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^d$. Then the fractional Laplacian of u is analytically given as*

$$(-\Delta)^{\frac{\alpha}{2}} u(\mathbf{x}) = c_{\alpha, d} |\sigma|^{-\alpha} {}_1F_1\left(\frac{d+\alpha}{2}; \frac{d}{2}; -\sigma^{-2}|\mathbf{x} - \mathbf{x}_0|_2^2\right) \text{ for } \mathbf{x} \in \mathbb{R}^d, \alpha \geq 0, \quad (3.24)$$

where ${}_1F_1$ denotes the confluent hypergeometric function, and

$$c_{\alpha, d} = \frac{2^\alpha \Gamma\left(\frac{d+\alpha}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

For a set $S_{\text{center}} = \{\tilde{\mathbf{x}}_i\}_{i=1}^M$, we let

$$p_{\text{GRBF}, \tilde{\theta}}(\mathbf{x}) = \sum_{i=1}^M w_i \mathcal{N}(\tilde{\mathbf{x}}_i, \sigma_i^2 \mathbf{I})(\mathbf{x}), \quad (3.25)$$

where $0 \leq \omega_i \leq 1$ such that $\sum_{i=1}^M \omega_i = 1$, σ_i is the bandwidth at $\tilde{\mathbf{x}}_i$, and \mathcal{N} denotes the Normal distribution,

$$\mathcal{N}(\tilde{\mathbf{x}}_i, \sigma_i^2 \mathbf{I})(\mathbf{x}) = (2\pi)^{-d/2} \sigma_i^{-d} \exp\left(-\frac{|\mathbf{x} - \tilde{\mathbf{x}}_i|_2^2}{2\sigma_i^2}\right). \quad (3.26)$$

Here both ω_i and σ_i can be trainable parameters, which are included in $\tilde{\theta}$. Using Lemma 3.2, we obtain that

$$\begin{aligned} (-\Delta)^{\alpha/2} p_{\text{GRBF}, \tilde{\theta}}(\mathbf{x}) &= \sum_{i=1}^M w_i (-\Delta)^{\alpha/2} \mathcal{N}(\tilde{\mathbf{x}}_i, \sigma_i^2 \mathbf{I})(\mathbf{x}) \\ &= c_{\alpha, d} \pi^{-d/2} 2^{-\frac{d+\alpha}{2}} \sum_{i=1}^M w_i |\sigma_i|^{-(d+\alpha)} {}_1F_1\left(\frac{d+\alpha}{2}; \frac{d}{2}; -\frac{|\mathbf{x} - \tilde{\mathbf{x}}_i|_2^2}{2\sigma_i^2}\right). \end{aligned} \quad (3.27)$$

Consequently, the loss function (3.6) can be rewrite by

$$\begin{aligned} \tilde{L}(p_{\text{KRnet},\theta}, p_{\text{GRBF},\hat{\theta}}) &= \frac{1}{N} \sum_{i=1}^N \left(\mathcal{L}p_{\text{KRnet},\theta}(\mathbf{x}^i) - c_{\alpha,d} \pi^{-\frac{d}{2}} 2^{-\frac{d+\alpha}{2}} \sum_{j=1}^M w_j |\sigma_j|^{-(d+\alpha)} {}_1F_1 \left(\frac{d+\alpha}{2}; \frac{d}{2}; -\frac{|\mathbf{x}^i - \tilde{\mathbf{x}}_j|^2}{2\sigma_j^2} \right) \right)^2 \\ &\quad + \frac{\beta_m}{N} \sum_{i=1}^N \left(p_{\text{KRnet},\theta}(\mathbf{x}^i) - \sum_{j=1}^M w_j (-\Delta)^{\alpha/2} \mathcal{N}(\tilde{\mathbf{x}}_j, \sigma_j^2 \mathbf{I})(\mathbf{x}^i) \right)^2, \end{aligned} \quad (3.28)$$

where the integer-order derivatives in operator \mathcal{L} can be conducted via automatic differentiation.

It is seen that the fractional Laplacian of $p_{\text{GRBF},\hat{\theta}}$ is determined by the confluent hypergeometric function ${}_1F_1(\cdot)$. If we allow σ_i to be a trainable parameter, we need the derivative of ${}_1F_1$ which is

$$\frac{d}{dx} {}_1F_1 \left(\frac{d+\alpha}{2}; \frac{d}{2}; x \right) = \frac{d+\alpha}{d} {}_1F_1 \left(\frac{d+\alpha}{2} + 1; \frac{d}{2} + 1; x \right). \quad (3.29)$$

In general it is computationally expensive to evaluate the confluent hypergeometric function. Fortunately, only the one-dimensional hypergeometric function is needed. We then use piecewise Chebyshev polynomials to approximate the one-dimensional confluent hypergeometric function up to a desired accuracy, which can be done once for all at the preprocessing stage.

3.5 An adaptive strategy for the training process

3.5.1 Where do we need adaptivity

We pay particular attention to two components of the algorithm that are closely related to adaptivity: one is the training set S and the other one is the auxiliary model $p_{\text{GRBF},\hat{\theta}}$. In MCNF, we only consider adaptivity for the training set S while in GRBFNF we address the adaptivity for both S and the model $p_{\text{GRBF},\hat{\theta}}$.

If the modeling capability of $p_{\text{KRnet},\theta}$ is sufficient, the training set S determines the accuracy of $p_{\text{KRnet},\theta^*}$ because it defines the loss function for both MCNF and GRBFNF. For a fixed domain, the collocation points in S are often sampled from a uniform distribution, which is obviously not optimal especially for a high-dimensional problem. Note that without any prior knowledge it is not straightforward to define a properly truncated domain to generate samples for S . For S with uniform samples, the loss function (3.1) of MCNF can be regarded as a Monte Carlo approximation of the L_2 norm of the residual in terms of a Lebesgue measure on the computation domain. The accuracy of such a Monte Carlo approximation depends on the number of samples and the variance of residual $R_\theta(\mathbf{x})$. One way to reduce the variance is to choose collocation points in terms of another measure instead of the Lebesgue measure such that the residual $R_\theta(\mathbf{x})$ is more uniform in terms of \mathbf{x} . For example, the loss function (3.1) can be regarded as

$$L(p_{\text{KRnet},\theta}) := \frac{1}{N_S} \sum_{i=1}^{N_S} |R_\theta(\mathbf{x}^i)|^2 \approx \int_{\mathbb{R}^d} R_\theta^2(\mathbf{x}) \rho(\mathbf{x}) d\mathbf{x}, \quad (3.30)$$

where \mathbf{x}^i are samples from a PDF $\rho(\mathbf{x})$ with $\rho(\mathbf{x}) > 0$ for any $\mathbf{x} \in \mathbb{R}^d$. A straightforward choice for the PDF $\rho(\mathbf{x})$ is the solution $p(\mathbf{x})$ because the residual is large more likely in the region of high probability density. If more samples are selected in the region of high density and less samples in the region of low density, the residual $R_\theta(\mathbf{x})$ would be more evenly distributed such that the Monte Carlo approximation of the integral of $R_\theta^2(\mathbf{x})$ in equation (3.30) would have a smaller statistical error. By minimizing a better approximation of the integral of $R_\theta^2(\mathbf{x})$, a better θ^* would be obtained. Since $p(\mathbf{x})$ is unknown, we may sample its approximation $p_{\text{KRnet},\theta^*}$ to form a new training set S . This suggests an adaptive solver for $p_{\text{KRnet},\theta}$, where we update S and $p_{\text{KRnet},\theta^*}$ alternately.

The auxiliary model $p_{\text{GRBF},\hat{\theta}}$ as an alternative representation of $p_{\text{KRnet},\theta}$ can be regarded as a kernel density estimator (KDE) since $p_{\text{KRnet},\theta}$ is a PDF. Given a set of samples $\{\mathbf{x}_i\}$, a general adaptive multivariate KDE takes the form [34],

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K_{\mathbf{H}_i}(\mathbf{x} - \mathbf{x}_i), \quad (3.31)$$

where H_i is the bandwidth matrix and $K_{\mathbf{H}_i} = |\mathbf{H}_i|^{-1} K(\mathbf{H}_i^{-1} \mathbf{x})$ rescales a kernel function $K(\mathbf{x})$. Due to Lemma 3.2, we choose $\mathbf{K}(\mathbf{x})$ as a standard multivariate Gaussian and $\mathbf{H}_i = h_i \mathbf{I}$ with h_i being the bandwidth shared by all

dimensions. An optimal bandwidth can be estimated either analytically or statistically. The main difference between $p_{\text{GRBF},\hat{\theta}}$ and a kernel density estimator is that the points in S_{center} may not be samples from the probability density function to be approximated. This is why $p_{\text{GRBF},\hat{\theta}}$ in equation (3.25) has variable coefficients w_i while the KDE in equation (3.31) has a constant coefficient $\frac{1}{N}$. Since $p_{\text{GRBF},\hat{\theta}^*} \approx p_{\text{KRnet},\theta^*}$, we expect that S_{center} has a data distribution that is consistent with $p_{\text{KRnet},\theta^*}$. When $p_{\text{KRnet},\theta^*}$ is updated adaptively, the set S_{center} should be updated accordingly for a more effective representation of $p_{\text{GRBF},\hat{\theta}^*}$. As $N \rightarrow \infty$, the KDE is simply the Monte Carlo simulation. However, for a GRBF approximation with a relatively small number of basis functions, varying w_i rather than the constant $\frac{1}{N}$ yield a better performance. Once a new S_{center} is specified, a straightforward idea to update the parameters of GRBFs is to project $p_{\text{KRnet},\theta^*}$ onto the new space spanned by the Gaussian radial basis functions with updated centers.

3.5.2 Adaptivity of MCNF

We propose the following adaptive sampling strategy to update the training set S . The initial collocation points in S are drawn from a uniform distribution in an area determined by our prior knowledge of $p(\mathbf{x})$. Then we solve the optimization problem (3.24) via the Adam optimizer to obtain optimal θ^* , which corresponds to a NF mapping $f_{\theta^{*,0}}$ and a PDF $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x})$. We subsequently update S using samples from $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x})$. To be precise, we sample the latent Gaussian random variable \mathbf{Z} , and use the samples of $\mathbf{X} = (f_{\theta^{*,0}})^{-1}(\mathbf{Z})$ to form the new training set S_1 . With S_1 , we start a new round of training to update $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x})$. We repeat this procedure until the maximum iteration number is reached. Such a strategy can be concluded as follows.

1. Generate an initial training set with samples uniformly distributed in $\Omega_0 \subset \mathbb{R}^d$:

$$S_0 = \{\mathbf{x}^{i,0}\}_{i=1}^{N_S} \subset \Omega_0, \quad \mathbf{x}^{i,0} \sim \text{Uniform } \Omega_0.$$

2. Train the KRnet by minimizing the loss function (3.23) with training data S_0 and hyper-parameter r_ϵ, r_0 to obtain $\theta^{*,0}$.

$$\theta^{*,0} = \arg \min_{\theta} \hat{L}(p_{\text{KRnet},\theta}; r_\epsilon, r_0).$$

3. Generate samples from $p_{\text{KRnet},\theta^{*,0}}(\cdot)$ to get a new training set $S_1 = \{\mathbf{x}^{i,1}\}_{i=1}^{N_S}$, and set $S_0 = S_1$. Notice that $\mathbf{x}^{i,1}$ can be obtained by transforming the prior Gaussian samples via the inverse temporal normalizing flow,

$$\mathbf{z}^{i,1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{x}^{i,1} = (f_{\theta^{*,0}})^{-1}(\mathbf{z}^{i,1}).$$

4. Repeat steps 2-3 for N_{adaptive} times to get a convergent approximation.

The algorithm for MCNF is given in Algorithm 1 and the flow chart is given in figure 2. Mini batches are used to accelerate the training process. Since the initial training points are uniformly distributed, we only expect that $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x})$ could capture the main behavior of the exact solution $p(\mathbf{x})$, which implies that a relatively small number of epochs is enough. As the convergence is being established by the adaptive procedure, we expect that $p_{\text{KRnet},\theta^{*,i}}(\mathbf{x})$ could capture more details of $p(\mathbf{x})$ as the iteration number i increases, which implies that the number of epochs may increase accordingly. We introduce a hyper-parameter γ in Algorithm 1 to represent the growth rate of epoch number for each adaptivity iteration.

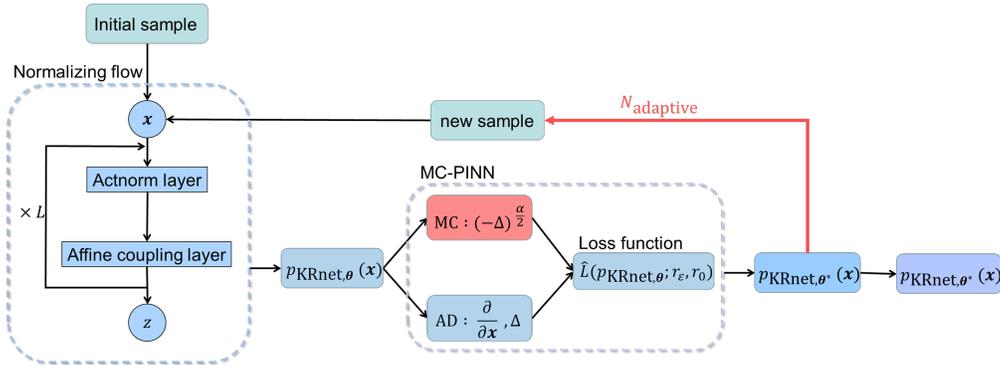


Figure 2: Flow chart of MCNF.

Algorithm 1 MCNF

Input: maximum epoch number N_e , maximum iteration number N_{adaptive} , fractional order α , hyper-parameter r_ϵ, r_0, γ , initial training data $S = \{\mathbf{x}^i\}_{i=1}^{N_S}$, tolerance ϵ_1, ϵ_2 ;
 $L_{\text{old}} = 0$;
for $k = 1, \dots, N_{\text{adaptive}}$ **do**
 for $j = 1, \dots, N_e$ **do**
 Divide S into m batch $\{S^{ib}\}_{ib=1}^m$ randomly;
 for $ib = 1, \dots, m$ **do**
 Compute the loss function (3.23) $\hat{L}^{ib}(p_{\text{KRnet}, \theta}; r_\epsilon, r_0)$ for mini-batch data S^{ib} and order α ;
 Update θ by using the Adam optimizer;
 $L_{\text{new}} = \frac{1}{m} \sum_{ib=1}^m \hat{L}^{ib}(p_{\text{KRnet}, \theta}; r_\epsilon, r_0)$;
 if $L_{\text{new}} < \epsilon_1$ or $|L_{\text{old}} - L_{\text{new}}| < \epsilon_2$ **then**
 Break;
 else
 $L_{\text{old}} = L_{\text{new}}$;
 $N_e = \gamma * N_e$;
 Sample from $p_{\text{KRnet}, \theta}(\cdot)$ and update training set S ;
 Output: The predicted solution $p_{\text{KRnet}, \theta}(\mathbf{x})$.

3.5.3 Adaptivity of GRBFNF

Compared to MCNF, we need to address the adaptivity for both S and the auxiliary model $p_{\text{GRBF}, \tilde{\theta}}$. The training set S follows the same adaptive procedure as in the MCNF. We here focus on the adaptivity for the auxiliary model. Depending on the prior knowledge, the initial center set S_{center} will be formed by uniform samples in a certain area. After $p_{\text{KRnet}, \theta^{*,0}}$ is obtained, S_{center} will be updated by samples from $p_{\text{KRnet}, \theta^{*,0}}$. To continue the training process with the updated S and S_{center} , we need to reinitialize the weights $\{w_i\}$ and the bandwidths $\{\sigma_i\}$ for GRBFs of the auxiliary model, which will be done by solving a least-square problem.

Reinitialization of the GRBFs. Once $p_{\text{KRnet}, \theta^{*,k}}$ is obtained for the k -th adaptivity iteration, we sample it to update the training set from S_k to S_{k+1} and GRBF centers from $S_{\text{center},k}$ to $S_{\text{center},k+1}$. The new auxiliary model is defined as

$$p_{\text{GRBF}, \tilde{\theta}^{k+1}}(\mathbf{x}) = \sum_{i=1}^M w_i N(\tilde{\mathbf{x}}_i^{k+1}, \sigma_i^2 \mathbf{I}_d)(\mathbf{x}), \quad \tilde{\mathbf{x}}_i^{k+1} \in S_{\text{center},k+1}. \quad (3.32)$$

and initialized as

$$\{w_{\text{new},i}, \sigma_{\text{new},i}\}_{i=1}^M = \arg \min_{w_i, \sigma_i} \text{Loss} = \arg \min_{w_i, \sigma_i} \frac{1}{N_{S_{k+1}}} \sum_{j=1}^{N_{S_{k+1}}} (p_{\text{GRBF}, \tilde{\theta}^{k+1}}(\mathbf{x}^j) - p_{\text{KRnet}, \theta^{*,k}}(\mathbf{x}^j))^2, \quad (3.33)$$

where the Adam optimizer is used to solve the above optimization problem. After initialization both the weights $\{w_i\}$ and the bandwidths $\{\sigma_i\}$ are trainable.

Such a strategy can be concluded as follows.

1. Generate initial training sets S and S_{center} with samples uniformly distributed in a certain physical domain:

$$S_0 = \{\mathbf{x}^{i,0}\}_{i=1}^{N_S} \subset \Omega_0, \quad \mathbf{x}^{i,0} \sim \text{Uniform } \Omega_0,$$

$$S_{\text{center},0} = \{\tilde{\mathbf{x}}_i^0\}_{i=1}^M \subset \Omega_0, \quad \tilde{\mathbf{x}}_i^0 \sim \text{Uniform } \Omega_0.$$

2. Train the KRnet by minimizing the loss function (3.23) with training data S_0 to obtain $\theta^{*,0}$ and $\tilde{\theta}^{*,0}$, i.e.,

$$\{\theta^{*,0}, \tilde{\theta}^{*,0}\} = \arg \min_{\theta, \tilde{\theta}} \tilde{L}(p_{\text{KRnet}, \theta}, p_{\text{GRBF}, \tilde{\theta}}).$$

3. Generate samples with $p_{\text{KRnet}, \theta^{*,0}}$ to get a new training set $S_1 = \{\mathbf{x}^{i,1}\}_{i=1}^{N_r}$, and center set $S_{\text{center},1} = \{\tilde{\mathbf{x}}_i^1\}$. Notice that $\mathbf{x}^{i,1}$ and $\tilde{\mathbf{x}}_j^1$ can be obtained by transforming the prior Gaussian samples via the inverse temporal normalizing flow.

$$\mathbf{z}^{i,1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{x}^{i,1} = (f_{\theta^{*,0}})^{-1}(\mathbf{z}^{i,1}),$$

$$\tilde{\mathbf{z}}_j^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \tilde{\mathbf{x}}_j^1 = (f_{\theta^{*,0}})^{-1}(\tilde{\mathbf{z}}_j^1).$$

4. Project $p_{\text{KRnet},\theta^{*,0}}$ onto the new GRBF space by solving problem (3.33). Set $S_0 = S_1, S_{\text{center},0} = S_{\text{center},1}$.
5. Repeat steps 2-3 for N_{adaptive} times to get a convergent approximation.

The algorithm for GRBFNF is summarized in Algorithm 2 and a flow chart is given in figure 3.

Algorithm 2 GRBFNF

Input: maximum epoch number N_e , maximum iteration number N_{adaptive} , fractional order α , hyper parameter γ , initial training data $S = \{\mathbf{x}^i\}_{i=1}^N$, center set $S_{\text{center}} = \{\tilde{\mathbf{x}}_i\}_{i=1}^M$, tolerance ϵ_1, ϵ_2 ;
 $L_{\text{old}} = 0$;
for $k = 1, \dots, N_{\text{adaptive}}$ **do**
 for $j = 1, \dots, N_e$ **do**
 Divide S into m batch $\{S^{ib}\}_{ib=1}^m$ randomly;
 for $ib = 1, \dots, m$ **do**
 Compute the loss function (3.28) $\tilde{L}^{ib}(p_{\text{KRnet},\theta}, p_{\text{GRBF},\tilde{\theta}})$ for mini-batch data S^{ib} and fractional order α ;
 Update $\theta, \tilde{\theta}$ by using the Adam optimizer;
 $L_{\text{new}} = \frac{1}{m} \sum_{ib=1}^m \tilde{L}^{ib}(p_{\text{KRnet},\theta}, p_{\text{GRBF},\tilde{\theta}})$;
 if $L_{\text{new}} < \epsilon_1$ or $|L_{\text{old}} - L_{\text{new}}| < \epsilon_2$ **then**
 Break;
 else
 $L_{\text{old}} = L_{\text{new}}$;
 $N_e = \gamma * N_e$;
 Sample from $p_{\text{KRnet},\theta}(\cdot)$ and update training set S, S_{center} ;
 Update $p_{\text{GRBF},\tilde{\theta}}$ by solving optimization problem (3.33).
 Output: The predicted solution $p_{\text{KRnet},\theta}(\mathbf{x})$.

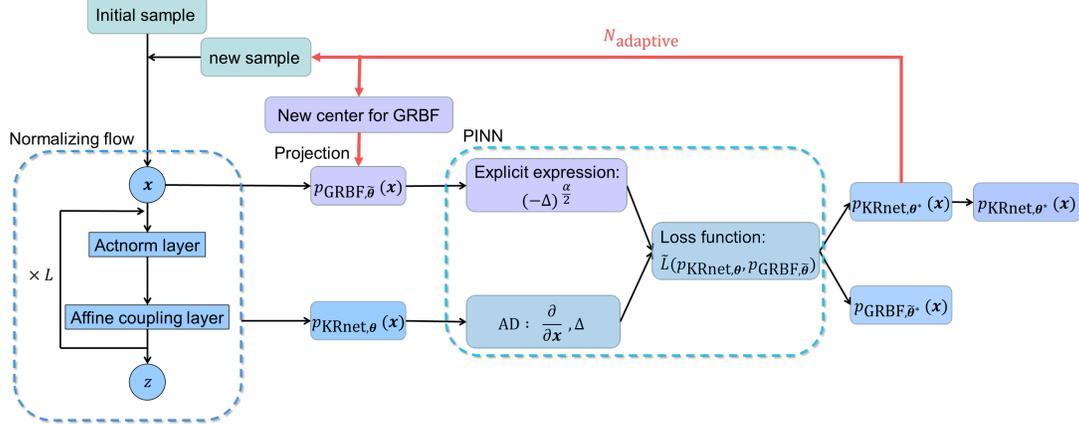


Figure 3: Flow chart of GRBFNF.

4 MCTNF for time-dependent fractional FPEs

The procedure is overall similar to the stationary case if we can address the time-dependent problems on a space-time domain. Considering that the update of GRBF centers cannot be straightforwardly generalized to the space-time domain, we only generalize MCNF for time-dependent FPEs in this work.

Given training sets $S_t = \{(x^i, t^i)\}_{i=1}^{N_t}$ and $S_{\text{ic}} = \{(\mathbf{x}_{\text{ic}}^i, p_0(\mathbf{x}_{\text{ic}}^i))\}_{i=1}^{N_{\text{ic}}}$ with $(\mathbf{x}, t) \in \mathbb{R}^d \times [0, T]$, we define the following loss function

$$L_T(p_{\text{KRnet},\theta}(\mathbf{x}, t)) := \frac{1}{N_t} \sum_{i=1}^{N_t} |R_{\theta}(\mathbf{x}^i, t^i)|^2 + \frac{\beta_D}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} |p_{\text{KRnet},\theta}(\mathbf{x}_{\text{ic}}^i, 0) - p_0(\mathbf{x}_{\text{ic}}^i)|^2, \quad (4.1)$$

where $p_0(\cdot)$ is an initial distribution, β_D is a weight parameter to balance the governing equation loss and the initial condition loss, and the residual $R_\theta(\mathbf{x}, t)$ is defined as

$$R_\theta(\mathbf{x}, t) := (\partial_t - \mathcal{L} + (-\Delta)^{\alpha/2})p_{\text{KRnet},\theta}(\mathbf{x}, t). \quad (4.2)$$

The optimal parameter θ^* can be obtained via solving the following optimization problem

$$\theta^* = \arg \min_{\theta} L_T(p_{\text{KRnet},\theta}). \quad (4.3)$$

Note that $p_{\text{KRnet},\theta}(\mathbf{x}, t)$ depends on both \mathbf{x} and t , meaning that the corresponding KRnet is a time-independent normalizing flow.

4.1 Time-dependent density model

The time-dependent PDF $p_{\text{KRnet},\theta}(\mathbf{x}, t)$ can be regarded as a conditional PDF $p_{\text{KRnet},\theta}(\mathbf{x}|t)$, which can be achieved by making the affine coupling layer time dependent. Let $\mathbf{x}^{[i]} = (\mathbf{x}^{[i],1}, \mathbf{x}^{[i],2})$ be a partition with $\mathbf{x}^{[i],1} \in \mathbb{R}^m$ and $\mathbf{x}^{[i],2} \in \mathbb{R}^{d-m}$. We define a time-dependent coupling layer $T_{\text{Aff},[i]}(\cdot, t)$ as follows:

$$\begin{aligned} \mathbf{x}^{[i],1} &= \mathbf{x}^{[i-1],1}, \\ \mathbf{x}^{[i],2} &= \mathbf{x}^{[i-1],2} \odot (\mathbf{1}_{d-m} + \beta \tanh(\mathbf{s}_{i,t}(\mathbf{x}^{[i-1],1}, t))) + e^{\zeta^i} \odot \tanh(\mathbf{q}_{i,t}(\mathbf{x}^{[i-1],1}, t)), \end{aligned} \quad (4.4)$$

where the only difference from the affine coupling layer defined in Section 3.2.2 is that $\mathbf{s}_{i,t}$ and $\mathbf{q}_{i,t}$ include t as their inputs such that

$$(\mathbf{s}_{i,t}, \mathbf{q}_{i,t}) = \text{NN}_{[i],t}(\mathbf{x}^{[i-1],1}, t). \quad (4.5)$$

Based on the Actnorm layer and time-dependent affine coupling layer, our simplified time-dependent KRnet can be represented by

$$\mathbf{z} = f_{\text{KRnet},\theta}(\mathbf{x}, t) = f_{[L]} \circ f_{[L-1]} \circ \cdots \circ f_{[1]}(\mathbf{x}, t), \quad (4.6)$$

$$f_{[i]} = T_{\text{Aff},[i]} \circ L_{\text{Actn},[i]}, \quad i = 1, \dots, L, \quad (4.7)$$

where $T_{\text{Aff},[i]}$ is a time-dependent affine coupling layer defined by equation (4.4) and $L_{\text{Actn},[i]}$ is an Actnorm layer defined by equation (3.11). For any t , we obtain an explicit condition PDF from equation (4.6)

$$p_{\text{KRnet},\theta}(\mathbf{x}, t) = p_{\text{KRnet},\theta}(\mathbf{x}|t) = p_{\mathbf{Z}}(f_{\text{KRnet},\theta}(\mathbf{x}, t)) |\nabla_{\mathbf{x}} f_{\text{KRnet},\theta}(\mathbf{x}, t)|. \quad (4.8)$$

Also note that for any t , $(-\Delta)^{\alpha/2} p_{\text{KRnet},\theta}(\mathbf{x}, t)$ can be approximated using the same procedure given in Section 3.3.

One commonly used strategy to enhance the effectiveness and robustness of the algorithm is to integrate some physical constraints explicitly into the algorithm. We here propose a simple modification for the affine coupling layer such that $p_{\text{KRnet},\theta}(\mathbf{x}, t)$ may satisfy the initial condition exactly without introducing a penalty term in the loss function.

Modified affine coupling layer. To include the initial condition, we consider a modified affine coupling layer $T_{\text{Aff}',[i]}(\cdot, t)$ as follows

$$\begin{aligned} \mathbf{x}^{[i],1} &= \mathbf{x}^{[i-1],1}, \\ \mathbf{x}^{[i],2} &= \mathbf{x}^{[i-1],2} \odot (\mathbf{1}_{d-m} + \beta \tanh(t\mathbf{s}_{i,t}(\mathbf{x}^{[i-1],1}, t))) + e^{\zeta^i} \odot \tanh(t\mathbf{q}_{i,t}(\mathbf{x}^{[i-1],1}, t)). \end{aligned}$$

where $\mathbf{s}_{i,t}$, $\mathbf{q}_{i,t}$ are modeled by neural network (4.5) and the only modification is the scaling of $\mathbf{s}_{i,t}$ and $\mathbf{q}_{i,t}$ with time t . Therefore $T_{\text{Aff}',[i]}$ is an identity when $t = 0$. Replacing $f_{[i]}$ with $T_{\text{Aff}',[i]}$ in equations (4.6) and (4.8) we obtain at $t = 0$,

$$\mathbf{z} = f_{\text{KRnet},\theta}(\mathbf{x}, 0) = \mathbf{x} \quad \text{or} \quad p_{\text{KRnet},\theta}(\mathbf{x}, 0) = p_{\mathbf{Z}}(\mathbf{x}). \quad (4.9)$$

If we choose the prior $p_{\mathbf{Z}}(\mathbf{z})$ the same as the initial distribution $p_0(\mathbf{z})$, the initial condition is satisfied exactly.

4.2 Adaptive procedure of MCTNF

We initialize $S_{t,0} = \{(\mathbf{x}^{i,0}, t^{i,0})\}$ using uniform samples from a space-time domain $\Omega_0 \times [0, T]$, where the volume Ω_0 is finite, and specify $S_{\text{ic},0} = \{(\mathbf{x}_{\text{ic}}^{i,0}, p_0(\mathbf{x}_{\text{ic}}^{i,0}))\}$. Then we solve the optimization problem (4.3) to obtain optimal $\theta^{*,0}$. After that we update training points $S_{t,0}$ and $S_{\text{ic},0}$ from $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x}, t)$. To be precise, we sample temporal points $\{t^{i,1}\}$ from a uniform distribution on $(0, T]$. For each $t^{i,1}$, we sample a latent normal random variable \mathbf{Z} to obtain a sample $\mathbf{x}^{i,1}$ of $\mathbf{X} = f_{\text{KRnet},\theta^{*,0}}^{-1}(\mathbf{Z}, t^{i,1})$. We then form $S_{t,1} = \{(\mathbf{x}^{i,1}, t^{i,1})\}$. $S_{\text{ic},1} = \{(\mathbf{x}_{\text{ic}}^{i,1})\}$ can be obtained via the same procedure by letting $t = 0$, i.e. $\mathbf{x}_{\text{ic}}^{i,1} = f_{\text{KRnet},\theta^{*,0}}^{-1}(\mathbf{z}^i, 0)$. We then continue the training process with $S_{t,1}$ and $S_{\text{ic},1}$. The procedure is repeated after the second training is done. Such a strategy can be concluded as follows.

1. Generate initial training sets using uniform samples on $\Omega_0 \times (0, T]$ where $\Omega_0 \in \mathbb{R}^d$ and $|\Omega_0| < \infty$:

$$S_{t,0} = \{(\mathbf{x}^{i,0}, t^{i,0})\}_{i=1}^{N_t}, \quad t^{i,0} \sim \text{Uniform}(0, T], \quad \mathbf{x}^{i,0} \sim \text{Uniform } \Omega_0,$$

$$S_{ic,0} = \{(\mathbf{x}_{ic}^{i,0}, p_0(\mathbf{x}_{ic}^{i,0}))\}, \quad \mathbf{x}_{ic}^{i,0} \sim \text{Uniform } \Omega_0.$$

2. Train the temporal KRnet by solving optimization problem (4.3) with training data $S_{t,0}, S_{ic,0}$ to obtain optimal parameters $\theta^{*,0}$:

$$\theta^{*,0} = \arg \min_{\theta} L_T(p_{\text{KRnet},\theta}(\mathbf{x}, t)).$$

3. Generate temporal samples from a uniform distribution on $(0, T]$ and spatial samples from $p_{\text{KRnet},\theta^{*,0}}(\mathbf{x}|t)$ to obtain $S_{t,1}, S_{ic,1}$.

$$S_{t,1} = \{(\mathbf{x}^{i,1}, t^{i,1})\}_{i=1}^{N_t}, \quad t^{i,1} \sim \text{Uniform}(0, T], \quad \mathbf{x}^{i,1} \sim p_{\text{KRnet},\theta^{*,0}}(\mathbf{x}|t = t^{i,1}),$$

$$S_{ic,1} = \{(\mathbf{x}_{ic}^{i,1}, p_0(\mathbf{x}_{ic}^{i,1}))\}, \quad \mathbf{x}_{ic}^{i,1} \sim p_{\text{KRnet},\theta^{*,0}}(\mathbf{x}|t = 0).$$

Set $S_{t,0} = S_{t,1}, S_{ic,0} = S_{ic,1}$.

4. Repeat steps 2-3 for N_{adaptive} times to get a convergent approximation.

Our algorithm for solving time-dependent fractional FPEs is given in Algorithm 3.

Algorithm 3 MCTNF

Input: maximum epoch number N_e , maximum iteration number N_{adaptive} , fractional order α , hyper-parameter r_ϵ, r_0, β_D , initial training data $S_t = \{(\mathbf{x}^i, t^i)\}_{i=1}^{N_t}, S_{ic} = \{(\mathbf{x}_{ic}^i, p_0(\mathbf{x}_{ic}^i))\}_{i=1}^{N_{ic}}, C_T = \{t_r^i\}_{i=1}^{N_r} \cup \{0\}_{i=1}^{N_{ic}}$, tolerance ϵ_1, ϵ_2 ;

$L_{old} = 0$;

for $k = 1, \dots, N_{\text{adaptive}}$ **do**

for $j = 1, \dots, N_e$ **do**

 Divide S_t, S_{ic} into m batches $\{S_t^{\text{ib}}\}_{\text{ib}=1}^m, \{S_{ic}^{\text{ib}}\}_{\text{ib}=1}^m$ randomly;

for $\text{ib} = 1, \dots, m$ **do**

 Compute the loss function $L_T(p_{\text{KRnet},\theta})$ for mini-batch data $S_t^{\text{ib}}, S_{ic}^{\text{ib}}$ and fractional order α ;

 Update θ_t by using the Adam optimizer;

$$L_{new} = \frac{1}{m} \sum_{\text{ib}=1}^m L_T(p_{\text{KRnet},\theta});$$

if $L_{new} < \epsilon_1$ or $|L_{old} - L_{new}| < \epsilon_2$ **then**

Break;

else

$$L_{old} = L_{new};$$

$N_e = \gamma * N_e$;

 Sample from $t \sim \text{Uniform}([0, T])$ and $p_{\text{KRnet},\theta_t}(\mathbf{x}|t)$ to update training sets S_t, S_{ic} ;

Output: The predicted solution $p_{\text{KRnet},\theta}(\mathbf{x}, t)$.

5 Numerical experiments

In this section, we present a series of comprehensive numerical tests to demonstrate the effectiveness of the proposed algorithms. To quantitatively evaluate the accuracy of the numerical solution $p_{\text{KRnet},\theta}$, we shall consider both the relative L_2 error $\|p^* - p_{\text{KRnet},\theta}\|_2 / \|p^*\|_2$ and the relative Kullback-Leibler (KL) divergence given by

$$\frac{D_{\text{KL}}(p^* || p_{\text{KRnet},\theta})}{H(p^*)} = \frac{\mathbb{E}_{p^*} [\log(p^*/p_{\text{KRnet},\theta})]}{-\mathbb{E}_{p^*} [\log p^*]},$$

where \mathbb{E} denotes the expectation and p^* the ground truth. We approximate the above relative L_2 error by Monte Carlo integration, namely,

$$\frac{\|p^* - p_{\text{KRnet},\theta}\|_2}{\|p^*\|_2} = \frac{(\int (p^*(\mathbf{x}) - p_{\text{KRnet},\theta}(\mathbf{x}))^2 d\mathbf{x})^{1/2}}{(\int (p^*(\mathbf{x}))^2 d\mathbf{x})^{1/2}} \approx \frac{(\sum_{i=1}^N (p^*(\mathbf{x}_i) - p_{\text{KRnet},\theta}(\mathbf{x}_i))^2 \Delta \mathbf{x}_i)^{1/2}}{(\sum_{i=1}^N p^*(\mathbf{x}_i)^2 \Delta \mathbf{x}_i)^{1/2}}.$$

Similarly, we also approximate the above relative KL divergence by Monte Carlo integration, i.e.,

$$\frac{D_{\text{KL}}(p^* || p_{\text{KRnet}, \theta})}{H(p^*)} \approx \frac{\sum_{i=1}^{N_v} (\log(p^*(\mathbf{x}_i) - \log p_{\text{KRnet}, \theta}(\mathbf{x}_i)))}{-\sum_{i=1}^{N_v} \log p^*(\mathbf{x}_i)}.$$

Here \mathbf{x}_i are drawn from the ground truth $p^*(\mathbf{x})$ and the amount of validation data is set to $N_v = 10^6$. An uniform mesh is used to calculate the relative L_2 error with mesh size 0.04 along each spatial dimension. For time-dependent problems, we obtain the relative L_2 error and the relative KL divergence according to the aforementioned formulas for each given t .

We shall employ hyperbolic tangent function (\tanh) as the activation function. For each i , $\text{NN}_{[i]}$ (see 3.16) is a feed forward neural network with two hidden layers. We use a half-half partition $\mathbf{x}_{[i]} = (\mathbf{x}_{[i],1}, \mathbf{x}_{[i],2})$, $\mathbf{x}_{[i],1} \in \mathbb{R}^{\lfloor d/2 \rfloor}$, $\mathbf{x}_{[i],2} \in \mathbb{R}^{d-\lfloor d/2 \rfloor}$ unless specified. We initialize all trainable parameters using Glorot initialization [13]. For the training procedure, we use the Adam optimizer [19]. All numerical tests are implemented with Pytorch.

5.1 FPE with only fractional Laplacian

We start with a toy example with only the fractional Laplacian term. Consider the following 2D equation

$$\begin{cases} (-\Delta)^{\alpha/2} p(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2, \\ \int_{\mathbb{R}^2} p(\mathbf{x}) d\mathbf{x} = 1, & p(\mathbf{x}) \geq 0, \end{cases} \quad (5.1)$$

where $f(\mathbf{x}) = -\frac{1}{2\pi} B(2, \alpha) 2^{-\frac{\alpha}{2}} \sigma^{-(2+\alpha)} {}_1F_1\left(\frac{2+\alpha}{2}; 1; -\frac{\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2\sigma^2}\right)$, $B(d, \alpha) = \frac{2^\alpha \Gamma((\alpha+d)/2)}{\Gamma(d/2)}$. The true solution is $p(\mathbf{x}) = \frac{1}{2\pi\sigma^2} \exp(-\frac{\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2\sigma^2})$. We take $\alpha = 1$, $\boldsymbol{\mu} = (1, 1)$, $\sigma = 2$.

For the NF, we take 8 affine coupling layers with 32 hidden neurons. The initial training set is generated via the uniform distributed points in $[0, 6]^2$. Note that $\mathbb{E}_p[1_{[0,6]^2}] \approx 0.5$, meaning that we have only used about 50% information about the effective domain of the target \mathbf{X} , where \mathbb{E}_p indicates the expectation with respect to $p(\mathbf{x})$ and 1_Ω is an indicator function for $\Omega \subset \mathbb{R}^2$. The sample size is 5000 and the batch size is 1024. Both MCNF and GRBFNF are applied. For the MCNF, the number of Monte Carlo samples used to approximate fractional Laplacian is 100, $r_0 = 4$, $r_\epsilon = 0.01$. The initial learning rate is 0.001 with half decay each 100 steps. For the GRBFNF, the number of basis functions is 100 and the initial center points of basis function are generated from a uniform distribution on $[0, 6]^2$. The learning rate is 0.01 with half decay each 300 steps.

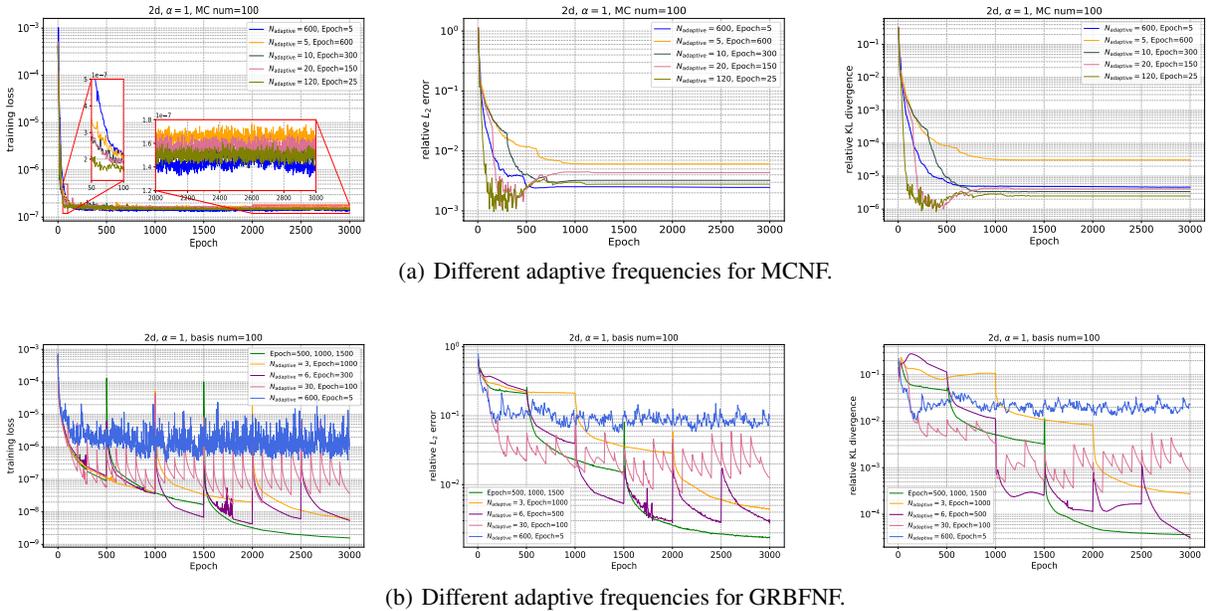


Figure 4: FPE with only fractional Laplacian. Left: Training loss. Middle: The relative L_2 error. Right: The relative KL divergence.

We first discuss the training strategy of MCNF and GRBFNF by adjusting the adaptive frequency of training. We present the training loss, relative L_2 error and relative KL divergence for different adaptive frequencies in Fig. 4. For the MCNF method, increasing the adaptive frequency leads to better results because the MC approximation of the fractional Laplacian is independent of the update of S . However, for the GRBFNF method, the adaptivity should not be activated until the current models is well trained, otherwise, the loss may be stuck in the transition period induced by the re-initialization of current models.

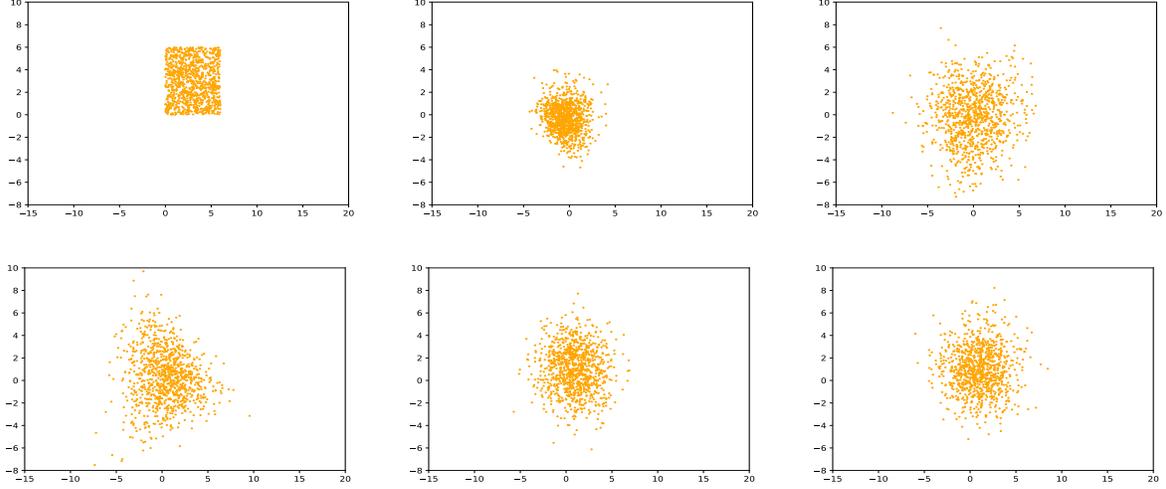
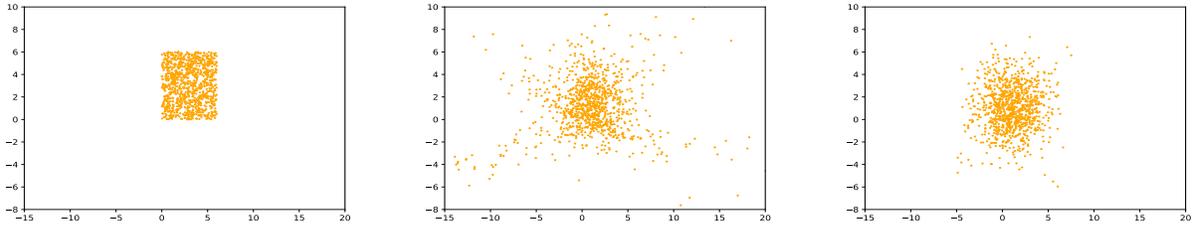
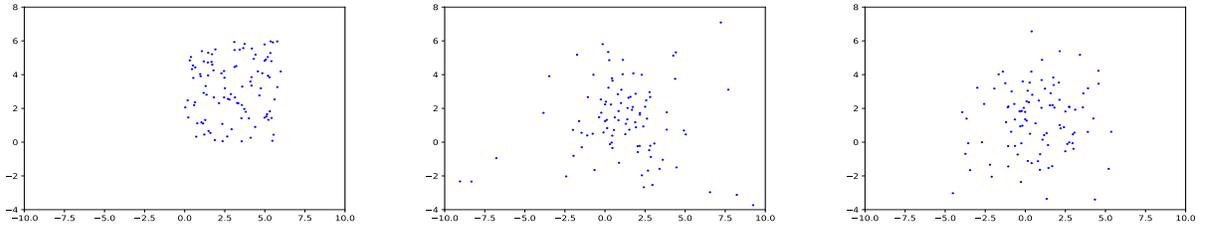


Figure 5: Distribution of training samples at different adaptivity iteration numbers in MCNF. From left to right and from top to bottom, $k = 1, 2, 3, 5, 30, 250$.



(a) Training samples at different adaptive iterations. From left to right, $k = 1, 2, 3$.



(b) Center points of GRBFs at different adaptive iterations. From left to right, $k = 1, 2, 3$.

Figure 6: Adaptivity of GRBFNF for the training set and the centers of GRBF basis functions.

Next, we focus on two experiments to investigate how adaptivity works. Specially, for the MCNF, we choose 600 adaptivity iterations with 5 epochs for each iteration. And for the GRBFNF, we choose 3 adaptivity iterations with increasing epochs, 500 epochs in the first adaptivity iteration, 1000 epochs in the second adaptivity iteration and 2000 epochs in the last adaptivity iteration. The time cost of MCNF and GRBFNF is 64 minutes, 46 minutes respectively.

The training points as well as center points of the basis functions for different adaptivity iteration numbers are presented in Fig. 5 and Fig. 6. One can clearly observe that the training points and center points of the basis functions become increasingly closer to the ground truth as the iteration number increases, showing that adaptive sampling scheme is effective.

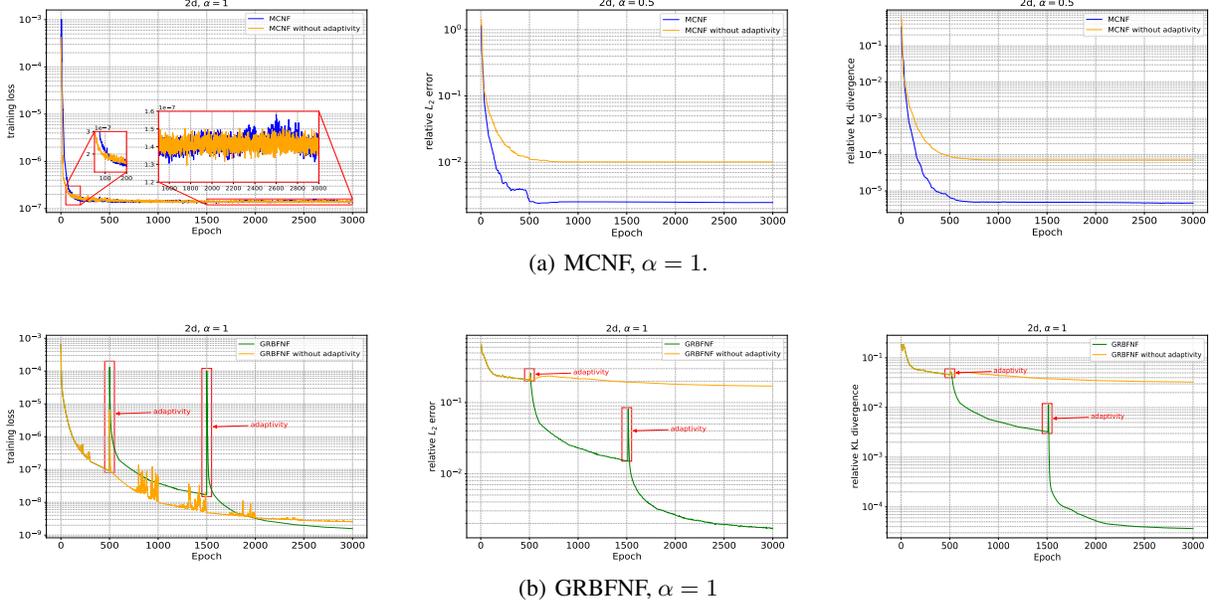


Figure 7: Comparison between adaptive and non-adaptive methods. Top row: MCNF. Bottom row: GRBFNF. Left: training loss. Middle: relative L_2 error. Right: relative KL divergence.

What's more, we compare our adaptive methods with non-adaptive methods in Fig. 7. It can be seen that, the accuracy of adaptive algorithm is higher than that of the non-adaptive algorithm especially for GRBFNF. The computational area of non-adaptive method is always $[0, 6]^2$, which certainly affects the performance outside this area. That is to say, the numerical solution can approximate the ground truth well inside predetermined area while fail to capture the information outside this area especially when the prior knowledge is not enough to design a suitable computational area. We drawn the ground truth in Fig. 8. The comparison between the predicted solution and the exact solution are presented in Fig. 9, from where we can clearly observe that the non-adaptive methods show larger errors in the area outside the computational area $[0, 6]^2$. On the other hand, our methods update the training points adaptively, which can effectively alleviate the limitation of a fixed computational area. Both the solutions of MCNF and GRBFNF yield excellent agreement with the exact solution. The relative L_2 error and the relative KL divergence with different adaptivity iteration numbers are also provided in Fig.10. The relative L_2 error of GRBFNF is smaller than MCNF while the relative KL divergence of GRBFNF is larger than MCNF.

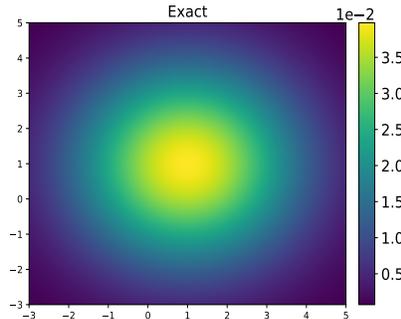


Figure 8: The reference solution of FFP with only the fractional Laplacian term.

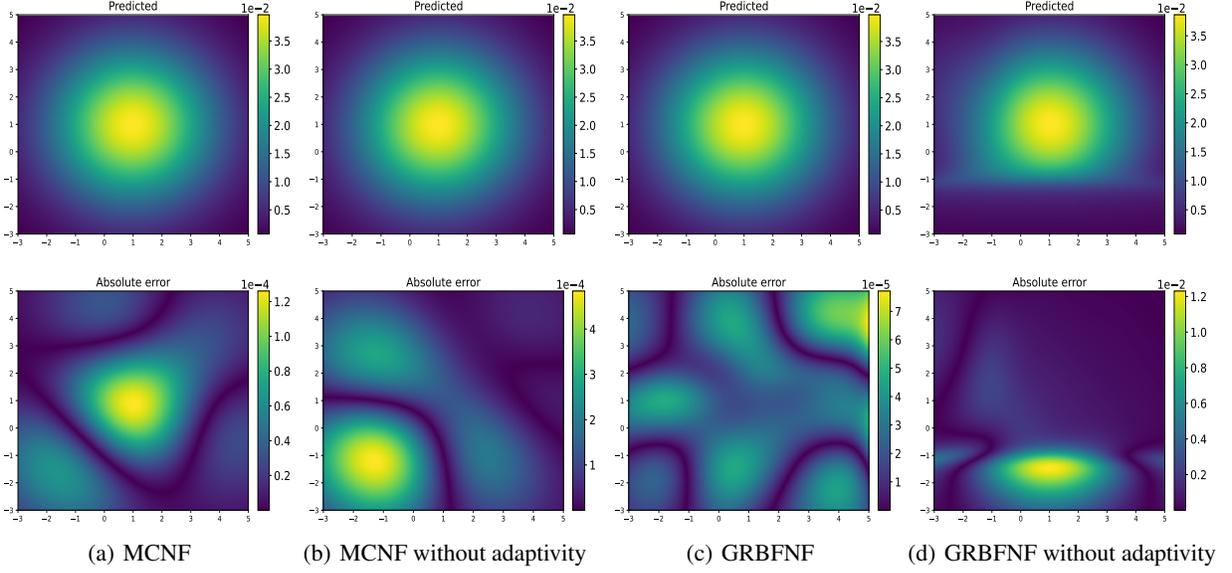


Figure 9: Comparison between the predicted solutions and the reference solutions. Top row: numerical solution. Bottom row: Absolute error between the numerical solution and the exact solution.

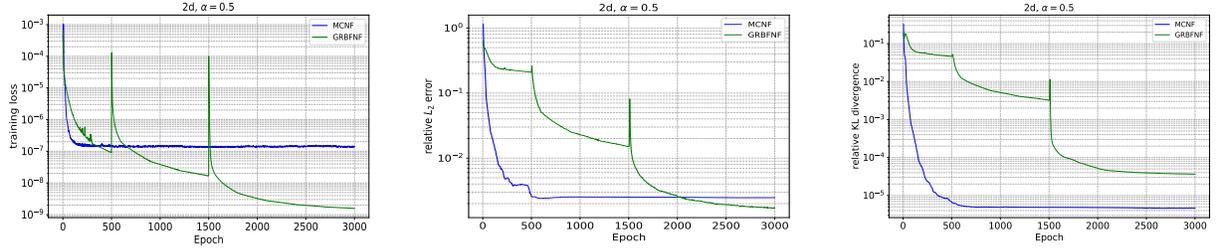


Figure 10: Convergence behavior of MCNF and GRBFNF. Left: training loss. Middle: relative L_2 error. Right: relative KL divergence.

Finally, we take $[-3, 3]^2$ to replace the above initial sampling area $[0, 6]^2$ and repeat the experiments to test the performance of MCNF and GRBFNF for fractional FPEs with different fractional order α . The results are presented in Fig. 11, where we also display the accuracy of Monte Carlo sampling method to compute associated fractional Laplacian. The numerical error of approximating fractional Laplacian is defined by $\frac{\sum_i |(-\Delta)^{\frac{\alpha}{2}} [p](\mathbf{x}_i) - \mathcal{M}[p](\mathbf{x}_i)|^2}{\sum_i |(-\Delta)^{\frac{\alpha}{2}} [p](\mathbf{x}_i)|^2}$ where $\mathcal{M}[p]$ denotes numerical approximation. Both MCNF and GRBFNF arrive good agreement with the ground truth for $\alpha = 0.5, 1, 1.5, 1.8$.

5.2 Bimodal distribution

To test the performance of MCNF and GRBFNF with respect to a bimodal distribution, we consider

$$\begin{cases} \nabla \cdot (\mathbf{g}(\mathbf{x})p(\mathbf{x})) + 0.05\Delta p(\mathbf{x}) - (-\Delta)^{\alpha/2}p(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2, \\ \int_{\mathbb{R}^2} p(\mathbf{x})d\mathbf{x} = 1, & p(\mathbf{x}) \geq 0, \end{cases} \quad (5.2)$$

where $\mathbf{g}(\mathbf{x}) = 0.2\mathbf{x}$,

$$f(\mathbf{x}) = \frac{1}{5\pi} \nabla \cdot (\exp(-2\|\mathbf{x} - \mathbf{1}_2\|_2^2)) - \frac{1}{\pi} B(2, \alpha) 2^{\frac{\alpha}{2}} \left({}_1F_1\left(\frac{2+\alpha}{2}; 1; -2\|\mathbf{x}\|_2^2\right) + {}_1F_1\left(\frac{2+\alpha}{2}; 1; -2\|\mathbf{x} - \mathbf{1}_2\|_2^2\right) \right).$$

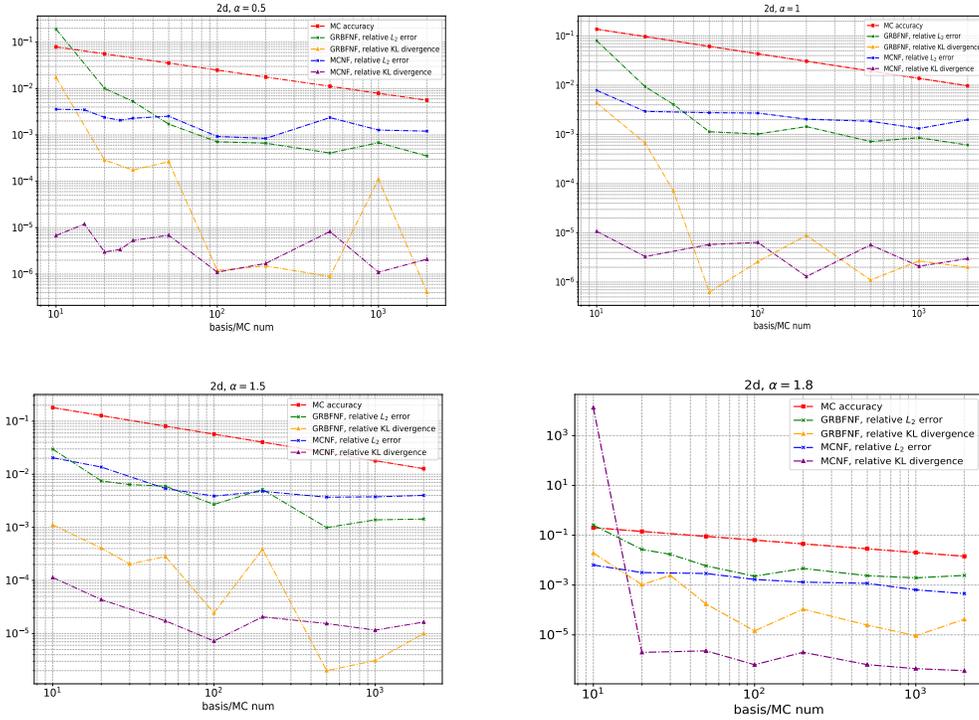


Figure 11: Error decay of MCNF and GRBFNF for different α in terms of the number of MC samples and GRBF basis functions. Left: relative L_2 error. Right: relative KL divergence.

The true solution is

$$p(\mathbf{x}) = \frac{1}{\pi} \left(\exp \left(-2\|\mathbf{x}\|_2^2 \right) + \exp \left(-2\|\mathbf{x} - \mathbf{1}_2\|_2^2 \right) \right).$$

For the NF, we take $L = 8$ affine coupling layers with 32 hidden neurons. The initial training set is generated via uniformly distributed points in $[-3, 3]^2$. The sample size is 5000 and the batch size is set to be 1024. Both MCNF and GRBFNF are applied. For the MCNF, the number of the samples used to approximate fractional Laplacian is 100, $r_0 = 0.3$, $r_\epsilon = 0.0001$. 800 adaptivity iterations with 5 epochs for each adaptivity iteration are conducted. The initial learning rate is 0.001 with 80% decay each 3000 steps. For the GRBFNF, the number of the basis functions is 100 and the initial center points of basis functions are generated from a uniform distribution in area $[-3, 3]$. 3 adaptivity iterations with increasing epochs are conducted for this problem, i.e. 500 epochs for the first adaptivity iteration, 1000 epochs for the second adaptivity iteration, and 2500 epochs for the last adaptive iteration. The learning rate is 0.01 with half decay every 300 steps and is reset to 0.005 after each adaptivity step.

We also apply MCNF and GRBFNF without adaptivity to solve this problem. The relative L_2 error and the relative KL divergence are provided in Fig. 12, which again verifies the strength of the adaptive methods. Although in this example, the unknown PDF is mainly concentrated in the initial sampling area which is different from the previous example, uniform samples used by non-adaptive methods fail to yield an accurate approximation and the adaptive sampling may improve the results by at least one order of magnitude. The exact solution is presented in Fig. 13. The comparison between the predicted solution and the exact solution are presented in Fig. 14. Both MCNF and GRBFNF can approximate the exact solution well. The training loss, the relative L_2 error and the relative KL divergence are presented in Fig. 15. The GRBFNF shows better performance than MCNF in this example.

5.3 High dimensional fractional Fokker-Planck equations

In this part, we consider a high-dimensional FPE.

$$\begin{cases} \nabla \cdot (\mathbf{g}(\mathbf{x})p(\mathbf{x})) + \Delta p(\mathbf{x}) - (-\Delta)^{\alpha/2} p(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^d, \\ \int_{\mathbb{R}^2} p(\mathbf{x}) d\mathbf{x} = 1, & p(\mathbf{x}) \geq 0. \end{cases} \quad (5.3)$$

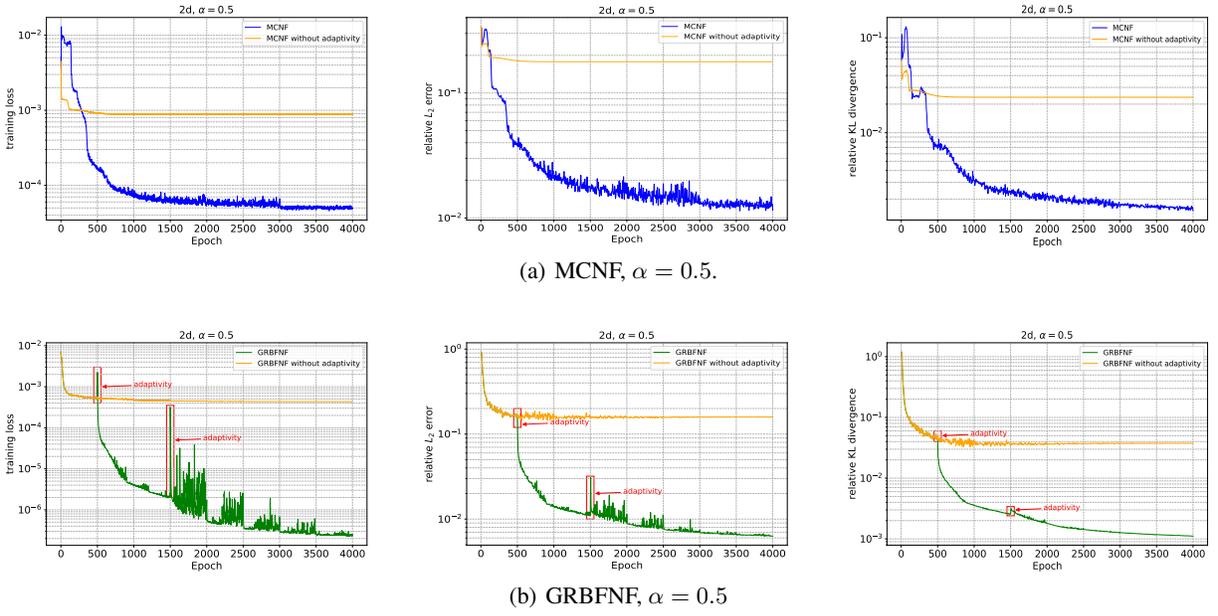


Figure 12: Convergence behavior of MCNF and GRBFNF with and without adaptive sampling. Left: training loss. Middle: relative L_2 error. Right: relative KL divergence.

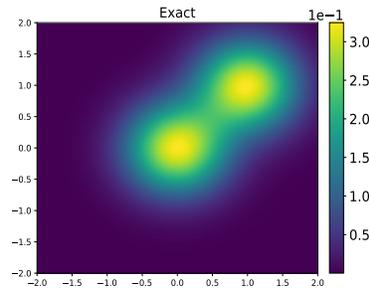


Figure 13: The ground truth of bimodal distribution.

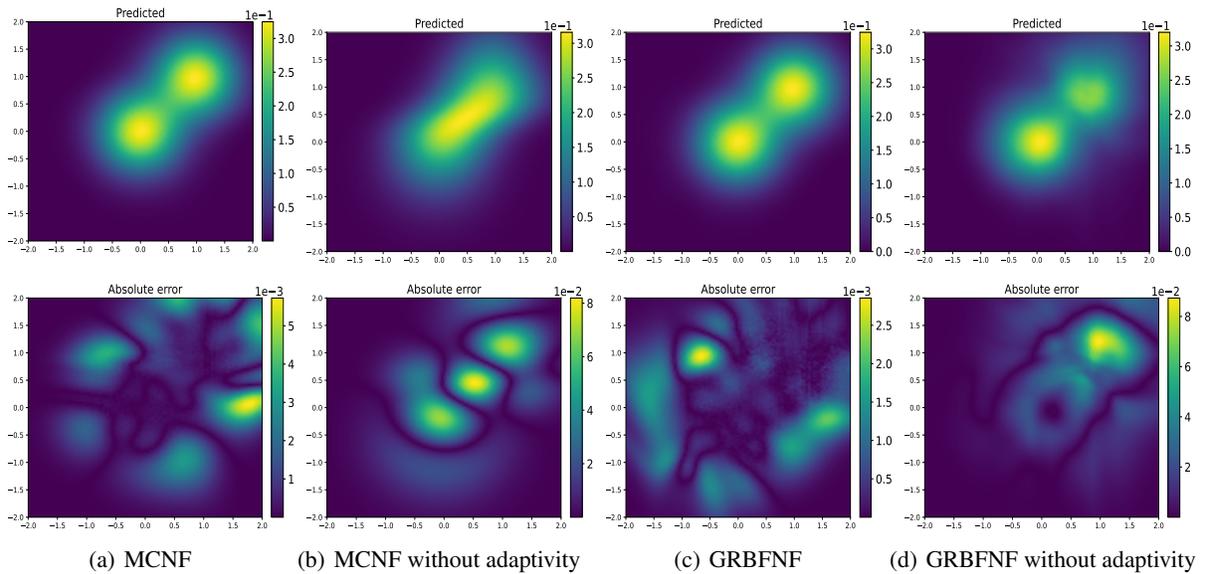


Figure 14: Comparison between the predicted solutions and the reference solutions. Top row: numerical solution. Bottom row: Absolute error between the numerical solution and the exact solution.

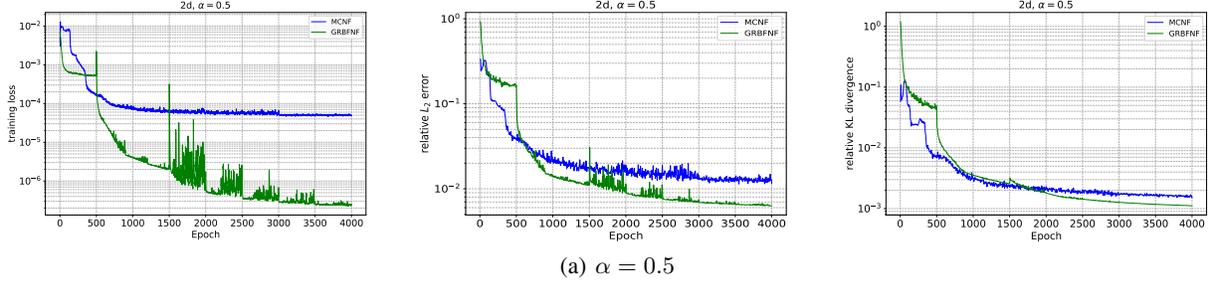


Figure 15: Convergence behavior of MCNF and GRBFNF for bimodal distribution. Left: training loss. Middle: relative L_2 error. Right: relative KL divergence. The unit of time is second.

where $g(\mathbf{x}) = \frac{\mathbf{x} - \boldsymbol{\mu}}{\sigma^2}$, the corresponding analytic solution is

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} \sigma^d} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right). \quad (5.4)$$

We take $d = 4, 6, 8$, $\boldsymbol{\mu} = \mathbf{1}_d$ and $\Sigma = \sigma^2 \mathbf{I}_d$, where \mathbf{I}_d is a d -dimensional identity matrix and $\sigma = 2$.

For high-dimensional problems, the PDF and the associated loss function may be too small, which results in numerical underflow. For the sake of numerical stability, we magnify the solution by multiplying a large enough constant C . Thus Cp satisfies

$$\frac{\partial(Cp)}{\partial t} = \mathcal{L}(Cp) - (-\Delta)^{\alpha/2}(Cp). \quad (5.5)$$

Actually, the C used here is 1 for $d = 4$, 10 for $d = 6$ and 200 for $d = 8$.

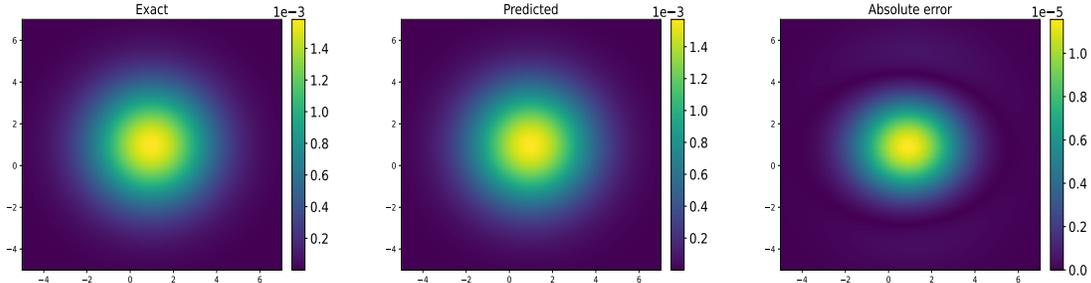


Figure 16: MCNF for 4-dimensional problem, where the first two dimensions are plotted at $x_3 = x_4 = 1$. Predicted solution versus the reference solution. Left: exact solution. Middle: prediction. Right: absolute error.

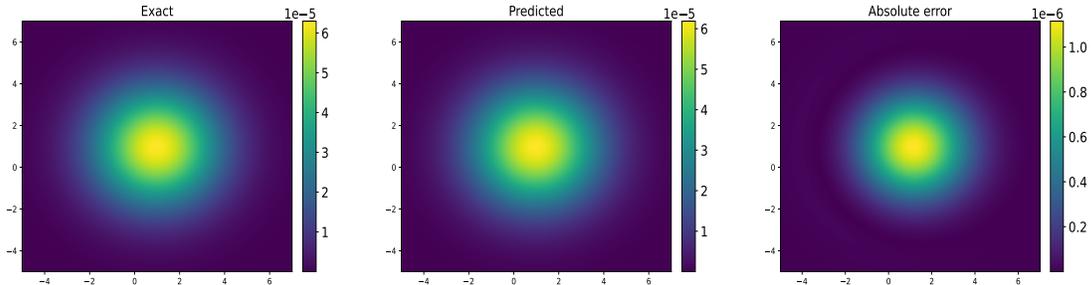


Figure 17: MCNF for 6-dimensional problem, where the first two dimensions are plotted at $x_3 = x_4 = x_5 = x_6 = 1$. Predicted solution versus the reference solution. Left: exact solution. Middle: prediction. Right: absolute error.

For NF, we take $L = 8$ affine coupling layers with 64 hidden neurons. The initial training set is generated via the uniform distributed points in $[-3, 5]^d$. The sample size is 50000. The batch size is set to be 4096. We employ MCNF

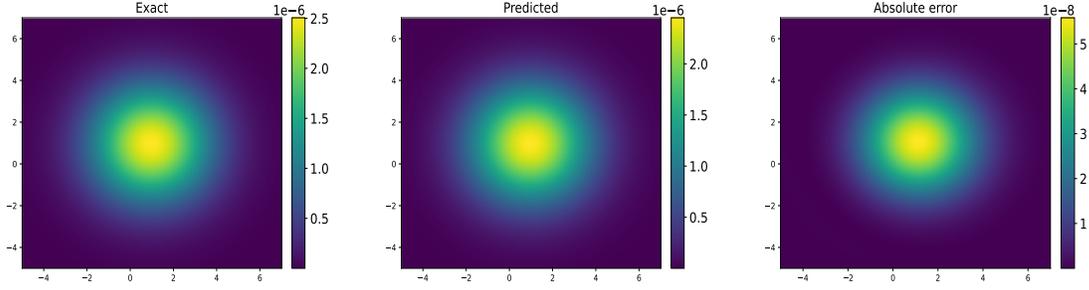


Figure 18: MCNF for the 8-dimensional problem, where the first two dimensions are plotted at $x_3 = x_4 = \dots = x_8 = 1$. Predicted solution versus the reference solution. Left: exact solution. Middle: prediction. Right: absolute error.

in this problem. GRBFNF is harder to train in high dimensional case since its structure is more complex. The number of samples used to approximate the fractional Laplacian is 200, $r_0 = 0.3$, $\epsilon = 0.0001$. We take half-half partition here. For $d = 4, 6, 100$ adaptivity iterations with 20 epochs for each adaptivity iteration are conducted. The learning rate is 0.001 with half decay each 300 steps. For $d = 8$, 20 adaptivity iterations with 200 epochs for each adaptivity iteration are conducted. The learning rate is 0.001 with half decay each 1000 steps. The comparisons between the MCNF solutions and the true solutions are presented in Fig. 16, Fig. 17, Fig. 18, which all show great performance of our approach. We also present the relative L_2 error and the relative KL divergence in Fig. 19.

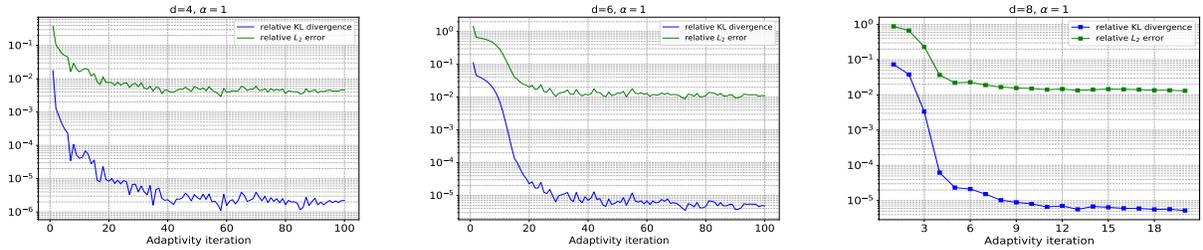


Figure 19: Convergence behavior of MCNF for high-dimensional FPEs. Left: $d = 4$. Middle: $d = 6$. Right: $d = 8$.

5.4 Time-dependent fractional FPE: Cauchy distribution

We consider the following stochastic process,

$$d\mathbf{X}_t = d\mathbf{L}_t^\alpha, \quad \alpha = 1. \quad (5.6)$$

For $d = 2$, the corresponding fractional Fokker-Planck equation is

$$\begin{aligned} \frac{\partial p}{\partial t} &= -(-\Delta)^{\alpha/2} p, \quad \alpha = 1, \\ p(\mathbf{x}, 0) &= p_0(\mathbf{x}). \end{aligned} \quad (5.7)$$

For the initial condition $p_0(\mathbf{x}) = \frac{1}{2\pi(1+\|\mathbf{x}\|_2^2)^{3/2}}$, the solution of (5.7) is $p(\mathbf{x}, t) = \frac{t+1}{2\pi((t+1)^2+\|\mathbf{x}\|_2^2)^{3/2}}$, where $\mathbf{x} \in \mathbb{R}^2$ and $t \in [0, 1]$.

For the NF, we take $L = 8$ affine coupling layers with 32 hidden neurons. The initial spatial samples are drawn from a uniform distribution in $[-3, 3]^2$ and temporal samples are generated from a uniform distribution in $[0, 1]$. The sample size is 100000 and the batch size is set to be 4096. For the MCNF, the number of samples used to approximate the fractional Laplacian is 100, $r_0 = 1$, $r_\epsilon = 0.01$. 100 adaptivity iterations with 5 epochs for each adaptivity iteration are conducted. The initial learning rate is 0.001 with half decay each 100 steps. One can observe a good agreement between the predicted solutions and the ground truth from the Fig. 20. The relative L_2 error and the relative KL divergence against time t for different adaptive iterations are also provided in Fig.21, which indicates the efficiency of adaptivity. We present the comparison of the relative error between the original MCTNF and modified MCTNF in Fig. 22. The modified MCTNF indeed improve the approximation. It is worth mentioning that the numerical error seems to increase as time evolves. We will explore this issue in the subsequent work.

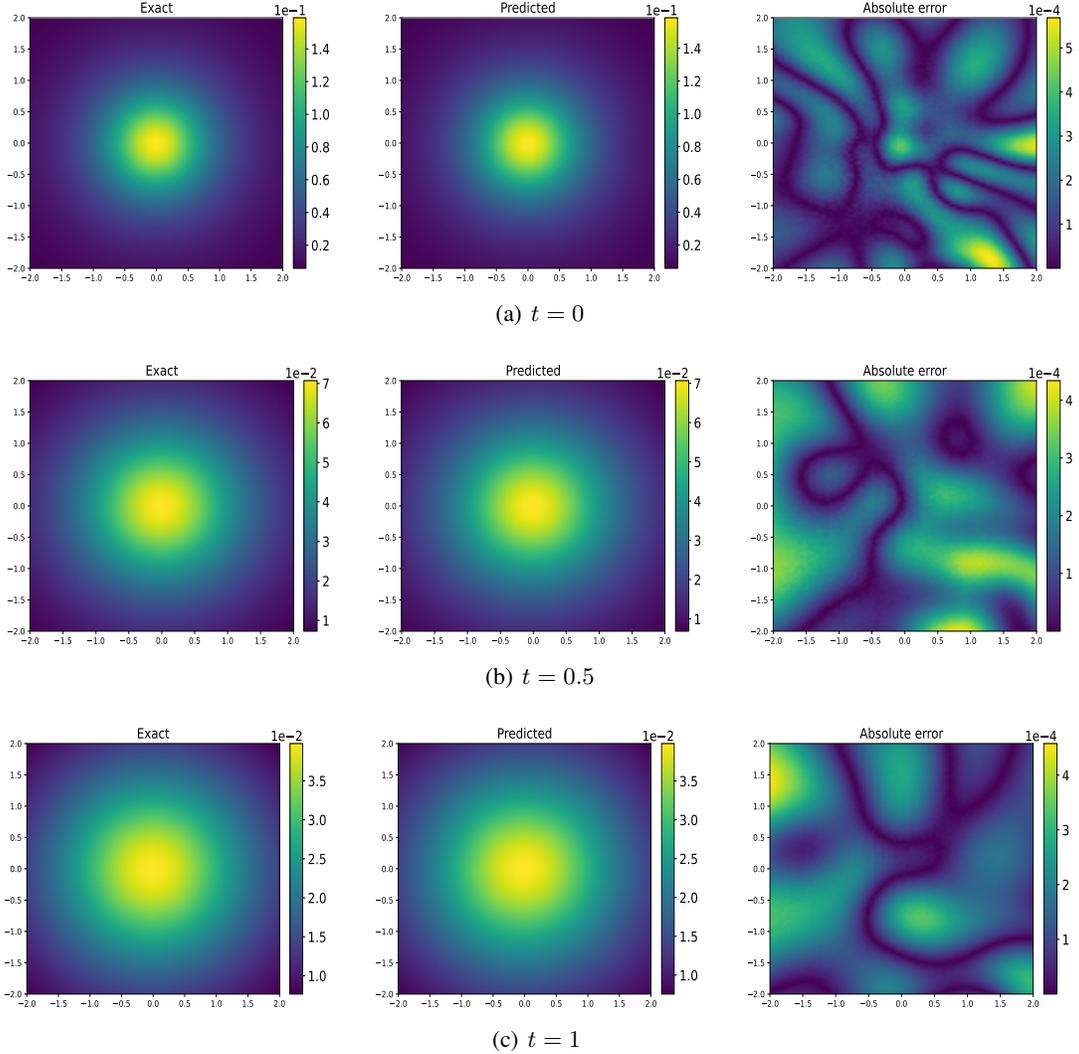
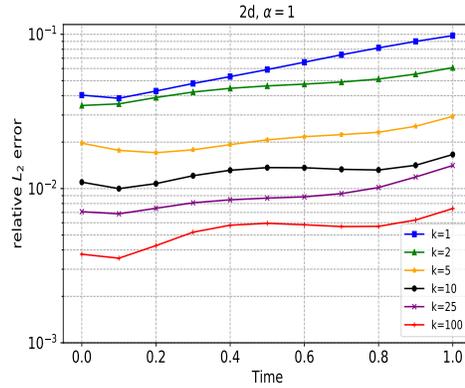
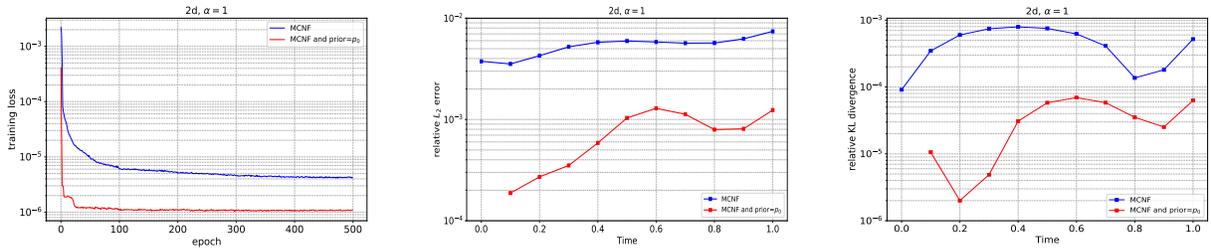


Figure 20: The predicted solutions versus the reference solutions for MCTNF at $t = 0, 0.5, 1$.

6 Conclusions

We have proposed flow-based adaptive algorithms for solving fractional FPEs. The core idea is to model the unknown PDF by a normalizing flow which yields an explicit PDF model as well as the corresponding exact random samples. For stationary FPEs, we proposed two methods: MCNF and GRBFNF. It is usually hard to choose a suitable computational area for unbounded problems. Our methods alleviate this difficulty by adaptively updating the training points. We train the MCNF model or GRBFNF model with current training points, and generate new training points using the current approximate solution. Then the training sets and the solution approximation are updated alternately. For time-dependent FPEs, we proposed MCTNF, where we modified the affine coupling layer to satisfy the initial condition exactly to improve the accuracy. Our approaches are validated by numerical experiments for both stationary and time-dependent FPEs. Compared to non-adaptive methods both MCNF and GRBFNF may improve the accuracy by at least one order of magnitude. From the numerical results, GRBFNF appeals to be more suitable for low-dimensional problems while MCNF demonstrates more flexibility for high-dimensional problems. The main difference between MCNF and GRBFNF is how the fractional Laplacian is approximated. MCNF uses the Monte Carlo approximation while GRBFNF relies on the GRBF approximation of the solution. GRBFNF is more effective for low-dimensional problems since the GRBF approximation is a linear model. MCNF performs better for high-dimensional problems because of the weak dependence of the Monte Carlo method on dimensionality. However, to further reduce the statistical error of the MC

Figure 21: The relative L_2 errors of MCTNF.Figure 22: Comparison between the original MCTNF and the modified MCTNF. Left: training loss. Middle: relative L_2 error. Right panel: relative KL divergence.

approximation of the fractional Laplacian, we may consider variance reduction techniques, which will be left for future study.

Acknowledgments

This work is supported by the National Key R&D Program of China (2020YFA0712000), the NSF of China (under grant numbers 12288201 and 11731006), and the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA25010404). The second author is supported by NSF grant DMS-1913163.

References

- [1] Nathalie Ayi, Maxime Herda, H el ene Hivert, and Isabelle Tristani. On a structure-preserving numerical method for fractional Fokker-Planck equations. [arXiv preprint arXiv:2107.13416](#), 2021.
- [2] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.
- [3] John Burkardt, Yixuan Wu, and Yanzhi Zhang. A unified meshfree pseudospectral method for solving both classical and fractional PDEs. *SIAM Journal on Scientific Computing*, 43(2):A1389–A1411, 2021.
- [4] Xiaoli Chen, Liu Yang, Jinqiao Duan, and George Em Karniadakis. Solving inverse stochastic problems from discrete particle observations using the Fokker–Planck equation and physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(3):B811–B830, 2021.
- [5] Weihua Deng. Finite element method for the space and time fractional Fokker–Planck equation. *SIAM journal on numerical analysis*, 47(1):204–226, 2009.
- [6] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. [arXiv preprint arXiv:1410.8516](#), 2014.

-
- [7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. [arXiv preprint arXiv:1605.08803](#), 2016.
- [8] Peter D Ditlevsen. Observation of α -stable noise induced millennial climate changes from an ice-core record. [Geophysical Research Letters](#), 26(10):1441–1444, 1999.
- [9] Weinan E and Bing Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. [Communications in Mathematics and Statistics](#), 6(1), 2018.
- [10] Michael B Elowitz, Arnold J Levine, Eric D Siggia, and Peter S Swain. Stochastic gene expression in a single cell. [Science](#), 297(5584):1183–1186, 2002.
- [11] Xiaodong Feng, Li Zeng, and Tao Zhou. Solving time dependent Fokker-Planck equations via temporal normalizing flow. [Commun. Comput. Phys.](#), pages 401–423, 2022.
- [12] Ting Gao, Jinqiao Duan, and Xiaofan Li. Fokker-Planck equations for stochastic dynamical systems with symmetric lévy motions. [Applied Mathematics and Computation](#), 278:1–20, 2016.
- [13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In [Proceedings of the thirteenth international conference on artificial intelligence and statistics](#), pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- [14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. [Advances in neural information processing systems](#), 27, 2014.
- [15] Ling Guo, Hao Wu, Xiaochen Yu, and Tao Zhou. Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations. [Computer Methods in Applied Mechanics and Engineering](#), 2022.
- [16] Ling Guo, Hao Wu, and Tao Zhou. Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models. [Journal of Computational Physics](#), 461:111202, 2022.
- [17] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. [Proceedings of the National Academy of Sciences](#), 115(34):8505–8510, 2018.
- [18] Raban Iten, Tony Metger, Henrik Wilming, Lída Del Rio, and Renato Renner. Discovering physical concepts with neural networks. [Physical review letters](#), 124(1):010508, 2020.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. [arXiv preprint arXiv:1412.6980](#), 2014.
- [20] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. [arXiv preprint arXiv:1807.03039](#), 2018.
- [21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. [arXiv preprint arXiv:1312.6114](#), 2013.
- [22] Shu Liu, Wuchen Li, Hongyuan Zha, and Haomin Zhou. Neural parametric Fokker-Planck equations. [SIAM Journal on Numerical Analysis](#), 60(3):1385–1449, 2022.
- [23] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. [Journal of Computational Physics](#), 401:109020, 2020.
- [24] Guofei Pang, Lu Lu, and George Karniadakis. fPINNs: Fractional physics-informed neural networks. [SIAM Journal on Scientific Computing](#), 41:A2603–A2626, 01 2019.
- [25] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. [Journal of Machine Learning Research](#), 22:1–64, 2021.
- [26] Tong Qin, Zhen Chen, John D Jakeman, and Dongbin Xiu. Deep learning of parameterized equations with applications to uncertainty quantification. [International Journal for Uncertainty Quantification](#), 11(2), 2021.
- [27] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. [Journal of Computational Physics](#), 378:686–707, 2019.
- [28] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. [Science](#), 367(6481):1026–1030, 2020.
- [29] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In [International conference on machine learning](#), pages 1530–1538. PMLR, 2015.
- [30] M.F. Shlesinger, G.M. Zaslavsky, and U. Frisch. Lévy Flights and Related Topics in Physics. 1995.

-
- [31] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [32] Keju Tang, Xiaoliang Wan, and Qifeng Liao. Deep density estimation via invertible block-triangular mapping. *Theoretical and Applied Mechanics Letters*, 10(3):143–148, 2020.
- [33] Kejun Tang, Xiaoliang Wan, and Qifeng Liao. Adaptive deep density approximation for Fokker-Planck equations. *Journal of Computational Physics*, 457:111080, 2022.
- [34] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.
- [35] Yong Xu, Wanrong Zan, Wantao Jia, and Jürgen Kurths. Path integral solutions of the governing equation of SDEs excited by Lévy white noise. *Journal of Computational Physics*, 394:41–55, 2019. ISSN 0021-9991.
- [36] Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing*, 42(1):A292–A317, 2020.
- [37] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.
- [38] Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.
- [39] Wanrong Zan, Yong Xu, Jürgen Kurths, Aleksei Chechkin, and Ralf Metzler. Stochastic dynamics driven by combined Lévy-Gaussian noise: Fractional Fokker-Planck-Kolmogorov equation and solution. *Journal of Physics A: Mathematical and Theoretical*, 53, 07 2020.
- [40] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.
- [41] Hao Zhang, Yong Xu, Yongge Li, and Jürgen Kurths. Statistical solution to SDEs with α -stable Lévy noise via deep neural network. *International Journal of Dynamics and Control*, 8(4):1129–1140, 2020.
- [42] Hui Zhang, Xiaoyun Jiang, and Xiu Yang. A time-space spectral method for the time-space fractional Fokker-Planck equation and its inverse problem. *Applied Mathematics and Computation*, 320:302–318, 2018.
- [43] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14):143001, 2018.
- [44] Yin hao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.